VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Vytvoření aplikace pro inventarizaci na platformě Microsoft Silverlight

Application for Inventarisation on Microsoft Silverlight

2013 Marek Slanina

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Marek Slanina**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Vytvoření aplikace pro inventarizaci na platformě Microsoft Silverlight
Application for Inventarisation on Microsoft Silverlight

Zásady pro vypracování:

Cílem práce je vytvoření aplikace pro inventarizaci majetku na platformě Microsoft Silverlight. Aplikace má za úkol přečíst informace o zařízení z QR kódu, v případě on-line přístupu zobrazit informace o zařízení z databáze.

1. Možnosti QR kódu.
2. Aplikace pro rozpoznání a přečtení QR kódu.
3. Komunikace aplikace s databází.
4. Webové rozhraní pro přístup k databázi.

Seznam doporučené odborné literatury:

Judith Sansweet *Introducing the QR Code: the Reality & the Magic: A QR Code primer*. ProofreadNZ Ltd (April 15, 2011). ISBN-10: 0473184516. ISBN-13: 978-0473184513.

Laurence Moroney. *Microsoft Silverlight 4 Step by Step* Publisher: Microsoft Press; Pap/Cdr edition (July 7, 2010). ISBN-10: 073563887X. ISBN-13: 978-0735638877.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 07.05.2013

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2013 .....

Děkuji svému vedoucímu práce Ing. Janu Skapovi, Ph.D. za poskytnuté konzultace a rady pro psaní této diplomové práce.

Abstrakt

Tato práce popisuje vývoj systému pro inventarizaci majetku s ukládáním dat pro identifikaci do QR kódu a ukládáním dalších dat do databáze. V práci je popsána vyvinutá aplikace pro čtení QR kódu, návrh a realizace databáze a také vytvoření webového rozhraní pro přístup k databázi.

Klíčová slova: inventarizace, QR kód, Microsoft Silverlight, ASP.NET MVC, Microsoft SQL Server

Abstract

This thesis describes development of inventory system with storing data for identification in a QR code and storing other data into the database. The thesis describes the developed application for reading QR codes, design and implementation of the database and also creation of web interface for database access.

Keywords: inventory, QR code, Microsoft Silverlight, ASP.NET MVC, Microsoft SQL Server

Seznam použitých zkratek a symbolů

ASP	–	Active Server Pages
CRUD	–	Creat Read Update Delete
CSV	–	Comma-separated values
DBML	–	Database Markup Language
HTML	–	HyperText Markup Language
ISO	–	International Organization for Standardization
LDAP	–	Lightweight Directory Access Protocol
LINQ	–	Language-Integrated Query
MVC	–	Model View Controller
ODT	–	OpenDocument
PNG	–	Portable Network Graphics
QR	–	Quick Response
RIA	–	Rich Internet application
SDK	–	Software development kit
SQL	–	Structured Query Language
SŘBD	–	Systém řízení báze dat
SSO	–	Single sign-on
T-SQL	–	Transactional Structured Query Language
URL	–	Uniform Resource Locator
XAML	–	Extensible Application Markup Language
ZXing	–	Zebra Crossing

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Katedra telekomunikačních technologií využívá ve svých laboratořích přístroje, vybavení a různé další předměty. Ty mohou být přemisťovány mezi laboratořemi, čímž může dojít k jejich ztrátě. Následně může dojít k nálezu předmětu, který nelze identifikovat a není tedy možné jej vrátit na původní pracoviště. Vytvořením systému pro inventarizaci a označováním předmětů QR kódy je možné tomuto zabránit. Při nalezení předmětu bude stačit přečíst kód. V něm bude uvedeno, kam patří a následně je možné předmět vrátit.

Systém může být stejně tak použit pro uchování některých dokumentů. Například naskenovaných faktur či revizních zpráv k přístrojům. Při potřebě zjistit informace z takového dokumentu není nutné vyhledávat jeho papírovou podobu v kartotéce, ale stačí zobrazit naskenovaný dokument v systému. V neposlední řadě může být systém užitečný při inventuře, kdy se kontroluje veškerý evidovaný majetek kus po kuse. Není třeba opisovat údaje ze štítku na kusu majetku, ale stačí z něj načíst QR kód aplikací a podle něj se v systému automaticky dohledá detail daného kusu.

V této práci popisuji, jak vznik právě takového systému probíhal a jaké při tom byly použity technologie. Téma mne zaujalo především z důvodu praktického využití QR kódů, které se v poslední době začaly hojně používat k nejrůznějším účelům. Využívají se na označování výrobků, etiketách zásilkových služeb, ukládání webových adres při marketingu a na vizitkách, kde díky jejich použití není potřeba vizitku předávat, ale pouze ukázat osobě, která si vaše kontaktní údaje načte mobilním telefonem. Téma mne oslovilo také kvůli platformě Microsoft Silverlight. Aplikace vyvinuté s použitím této technologie není třeba instalovat do počítače a při hostování na internetové stránce jsou přístupné online. K jejich používání je tedy možné využít libovolný počítač s nainstalovaným pluginem v prohlížeči a připojením k internetu.

Práce je rozdělena na šest kapitol. První kapitolou je tento úvod. Ve druhé kapitole nazvané „Možnosti QR kódu“ zmiňuji něco o jejich historii a vzniku, jejich typech a variantách. Zabývám se tím, jak funguje jejich kódování a dekódování, tedy jakým způsobem se generují a čtou. A také jaké mohou mít limity a omezení v souvislosti s požadavky na ukládaná data pro inventární systém. Na konci této kapitoly popisuji knihovnu ZXing, která byla pro čtení a generování QR kódů v inventárním systému použita.

Další kapitola popisuje platformu Microsoft Silverlight a vývoj aplikace pro čtení QR kódu na této platformě. Začátek kapitoly je věnován platformě samotné. Pokračuje požadavky kladenými na aplikaci pro inventární systém a končí popisem výsledné aplikace, která dokáže přečíst data z QR kódu. Pro dekódování využívá aplikace již zmiňovanou knihovnu ZXing.

Data pro systém jsou ukládána v databázi a jejímu návrhu se věnuji ve čtvrté kapitole. Systém používá databázi Microsoft SQL Server Express a Framework LINQ to SQL pro objektově relační mapování. Na získání dat je vhodné použít návrhový vzor, který umožní případnou výměnu databáze za jinou. Tento vzor je rovněž popsán ve čtvrté kapitole.

Pátá kapitola popisuje webové rozhraní inventárního systému, které bylo vyvinuto frameworkem ASP.NET MVC. Jeho použití umožnilo oddělit datový model webového

rozhraní, jeho vzhled a jeho chování do samostatných komponent. V úvodu kapitoly popisuji framework samotný, později vývoj vlastního webového rozhraní s jeho použitím a nakonec je v kapitole popsáno, jak jsem integroval Silverlight aplikaci do tohoto webového rozhraní.

V závěrečné kapitole zhodnocuji výslednou aplikaci, systém a uvádím několik nápadů na budoucí úpravy, vylepšení a nové funkce. Ty mohou být námětem na další diplomové či bakalářské práce.

Systém byl tvořen především pro evidenci přístrojů a proto jsou evidované položky v systému označovány jako „devices“. Nicméně je systém univerzální a je možné jej použít pro inventarizaci libovolných předmětů. V textu někdy užívám slovo „přístroj“ a někdy „předmět“, vždy je tím ale myšlen objekt, který reprezentuje kus evidovaného majetku s přiděleným inventárním číslem.

2 Možnosti QR kódu

2.1 Co je to QR kód

QR kód je obrazový symbol obsahující data. Vynalezla jej japonská firma Denso Wave už v roce 1994 [1]. Až v posledních několika letech se však tyto kódy dostaly do širšího povědomí veřejnosti a to ve spojení s mobilními telefony, především s těmi s operačním systémem. Mnoho z nich se prodává s již předinstalovanou aplikací pro skenování QR kódů a do těch modelů, do kterých aplikaci nenainstaloval výrobce nebo distributor, si ji může uživatel nainstalovat sám.

QR je zkratka slovního spojení „Quick Response“ a tato zkratka vlastně i vyjadřuje tři hlavní cíle, kterých má tato technologie dosahovat:

- umožnit rychlé skenování ze všech stran
- disponovat velkou kapacitou pro uložení dat na malé ploše
- poskytovat možnosti neomezených přenosných informací [1]

2.2 Teorie QR kódu

Pro pochopení toho, jak technologie QR kódů funguje, uvádím několik odstavců z normy ISO IEC-18004-2006, ve které je specifikována:

„QR kód je symbol reprezentovaný dvojrozměrnou maticí zobrazenou jako čtverec. Skládá se z hranatých modulů umístěných po celé své ploše. Existují čtyři technicky odlišné druhy QR kódů:

- *QR Code Model 1 byl původní specifikací pro QR kód a je popsán v AIM International Symbology Specification 97-001.*
- *QR Code Model 2 byl rozšířenou formou symboliky s přidanými funkcemi (především přidáním zarovnávacích vzorů napomáhajících k navigaci ve velkých symbolech) a byly základem první edice ISO/IEC 18004.*
- *QR Code 2005 (základ druhého vydání ISO/IEC 18004) je velice podobný QR Code Model 2 a ve svém formátu se liší jen přidáním možnosti pro znaky zrcadlově převrácené, vytištěny negativně (světlé symboly na tmavém podkladu) a s možností určení alternativní výchozí znakové sady.*
- *Formát Micro QR Code je variantou QR Code 2005 se sníženým počtem režijních modulů a omezeným rozsahem velikostí. To umožňuje uložit do symbolu malé až střední množství dat a je vhodné především pro přímé značení na součástky, komponenty a k aplikování tam, kde je prostor pro symboly silně omezen.*

QR Code 2005 obsahuje kromě hranatých modulů také unikátní vyhledávací vzory umístěné ve třech rozích kódu (v případě verze Micro QR Code jen v jednom rohu), které napomáhají jednoduššímu určení pozice, velikosti a sklonu kódu. Má stanoven širokou škálu velikostí a čtyři úrovně korekce chyb. Rozměry modulů jsou dány uživatelem, aby umožnily vytvářet symboly

různými technikami. QR Code Model 2 je plně kompatibilní s čtecími systémy verze QR Code 2005. Kódy Model 1 jsou doporučovány pouze pro použití v uzavřených systémech. Kódy ve verzi QR Code 2005 jsou doporučovány pro nové a otevřené systémy" [2].

Jak je vidět, není jen jeden druh QR kódu. Pro použití si tedy mohu vybrat ten, který bude požadavkům na systém vyhovovat. Ukázky jednotlivých druhů jsou vidět na obrázcích. Model 1 je zachycen na obrázku 1. Model 2 na obrázku 2 se od prvního liší přítomností menšího zarovnávacího vzoru v pravém dolním rohu. Obě ukázky symbolu jsou z webových stránek firmy Denso [3]. QR Code 2005 podporuje čtení zrcadlově převrácených symbolů a symbolů s invertovanými barvami. Jejich příklad uvádím na obrázku 3, který byl převzat z dokumentu ISO [2]. Je na něm vidět kód ve standardním zobrazení, s invertovanými barvami, kód zrcadlově převrácený a nakonec zrcadlově převrácený kód s invertovanými barvami. Pro úplnost uvádím ještě příklad Micro QR na obrázku 4 rovněž převzatý z webových stránek firmy Denso [3]. Dále v textu je již popsán pouze QR Code Model 2 (případně QR Code 2005) a srovnání s Micro QR.

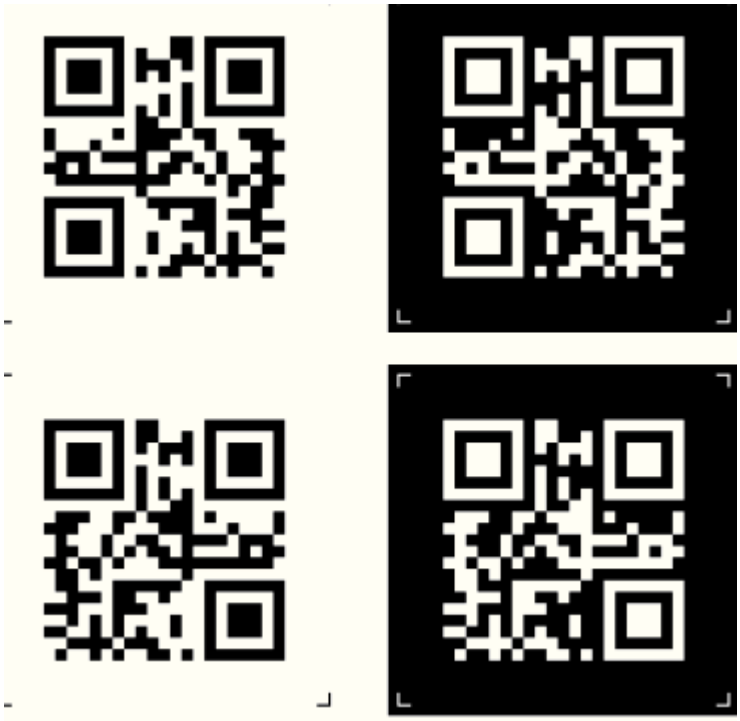
Obrázek 1: QR Model 1

Obrázek 2: QR Model 2

Pozice modulů se čísluje od levého horního rohu a od nuly. Význam bajtů je zobrazen jako hexadecimální číslo.

Základní charakteristika druhů kódu:

- QR kód poskytuje celou škálu funkcí a maximální datovou kapacitu



Obrázek 3: QR Code 2005

Obrázek 4: Micro QR Code

- Micro QR kód má naproti tomu kvůli redukované režii některá omezení schopností a datové kapacity

K dispozici jsou čtyři možnosti kódování znakových sad:

- Numerická data (číslice 0 – 9)
- Alfanumerická data (číslice 0 – 9; velká písmena A – Z; devět dalších znaků: mezera, $ % * + - . / :)
- Data v bytech (výchozí kódování je ISO/IEC 8859-1, ale je možné použít i UTF-8)
- Kanji data – pro kódování japonských znaků

Tmavý modul představuje binární jedničku a světlý modul binární nulu. S rozšířením QR Code 2005 je však možné použít i negativní zobrazení světlého symbolu na tmavém podkladu, kde je jednička reprezentována světlým modulem a nula tmavým.

Velikost symbolů:

- QR Code – identifikované verzí 1 až 40, velikost 21x21 modulů až 177x177 (přírůstek po čtyřech modulech)
- Micro – identifikované verzí M1 až M4, velikost 11x11 modulů až 17x17 modulů (přírůstek po dvou modulech)

Všechny druhy QR kódů mají užitečnou vlastnost, která zajišťuje jejich čitelnost i v případě částečného poškození. Vlastnost je umožněna díky korekci chyby. Verze QR kódu se uvádí ve formátu V-E, kde V znamená číslo verze (1-40) a E značí úroveň korekce chyby (L, M, Q, H). U Micro QR kódu se uvádí verze formátem MV-E kde M je pro rozpoznání Micro QR formátu, V a E značí, stejně jako u QR kódů, číslo verze (1-4) a úroveň korekce chyby (s hodnotou L, M, Q). Informace o verzi kódu je v symbolu uvedena společně s bity pro zjištění úrovně korekce.

Použita je Reed-Solomonova korekce chyby a umožňuje obnovu dat v symbolu. Pro verzi Micro QR jsou k dispozici pouze úrovně L, M a Q. Navíc u verze M1 je kapacita natolik omezena, že umožňuje pouze detekci chyby a tedy ne korekci. Schopnost korekce je podle úrovně následující:

- L 7%
- M 15%
- Q 25%
- H 30%

Kapacita znaků symbolů

- Maximální velikost symbolu QR kódu je verze 40-L a může obsahovat:
  - Numerická data: 7089 znaků
  - Alfanumerická data: 4296 znaků
  - Data v bajtech: 2953 znaků
  - Kanji data: 1817 znaků
- Maximální velikost symbolu Micro QR kódu je verze M4-L a může obsahovat:
  - Numerická data: 35 znaků
  - Alfanumerická data: 21 znaků
  - Data v bajtech: 15 znaků
  - Kanji data: 9 znaků

Vzhledem k doporučení v normě o použití QR Code 2005 pro nové systémy jsem zvažoval jeho použití v mém systému. Od Model 2 se ale liší jen možností pro převrácené symboly a to není vlastnost, která by byla v systému potřeba. Rozhodl jsem se proto v systému použít QR Code Model 2, pro který existuje více již vyvinutých knihoven pro dekódování.

2.3 Analýza potřeb pro systém

Umístěním QR kódu na předmět je umožněna jeho snadnější a rychlejší identifikace v budoucnu. Každý kus vybavení laboratoře má již nyní přiřazeno své unikátní inventární číslo, které je tím pádem možné použít jako identifikátor v systému a dohledat podle něj informace o předmětu z databáze.

Kromě inventárního čísla může kód obsahovat i další doplňující informace. K jejich přečtení nemusí být využita jen aplikace tvořená v rámci této diplomové práce, ale je možné použít libovolný software pro osobní počítač nebo mobilní telefon. Na první pohled se tedy může zdát, že by bylo vhodné do kódu ukládat co nejvíce informací, aby se nemusela aplikace dotazovat na vše do databáze. Nebo také proto, abychom mohli většinu informací získat přečtením QR kódu jakoukoliv aplikací, která je toho schopna. Čím více informací do kódu budeme chtít ukládat, tím více modulů v něm bude obsaženo. To následně buď zvětší rozměry symbolu a bude obtížnější je umístit na malé předměty, nebo zmenší velikost modulů a tím i velikost výsledného symbolu. Symbol s malými rozměry modulů bude zase těžší číst zařízením s méně kvalitní kamerou.

Inspiroval jsem se zavedenou praxí a prozkoumal již použité štítky s QR kódy v budovách VŠB-TUO. Ty jsou tištěny na lesklou žlutou lepící fólii a nalepeny na školní vybavení. V těchto QR kódech je uloženo inventární číslo a název předmětu v jednom textovém řetězci. Pro oddělení údajů v řetězci je použita dvojice středníků a pro jeho zakončení trojice středníků. Výsledný kód má rozměry 29 x 29 modulů. Příklad takového štítku je na obrázku 5, ve kterém je uložen text „39814175;;Monitor 17" LCD YUSMART;;;". QR kód na těchto štítcích je Model 2. To jsem zjistil pokusem o jeho dekódování knihovnou ZXing, kterou popisuji v další části této kapitoly.

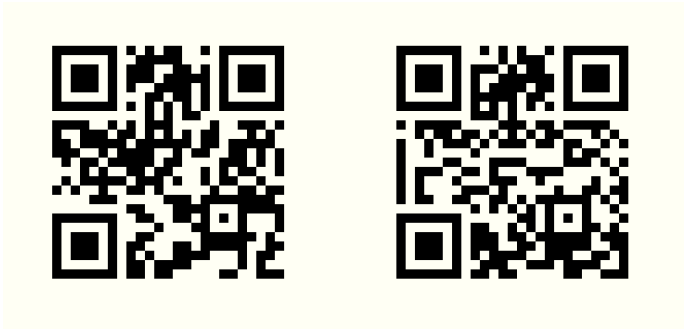
V případě mého systému jsem se s vedoucím práce shodl na tom, že QR kód musí obsahovat jednoznačnou identifikaci předmětu ve formě inventárního čísla a místnost, do které patří. Tyto údaje jsou nejčastěji potřeba a ostatní parametry, jako je název předmětu a jeho výrobce, se mohou načíst z databáze. Údaje budou v kódu uloženy ve formě textového řetězce a odděleny jedním středníkem. Aplikace pro čtení QR kódů, kterou jsem vyvinul a popisuji ji v další kapitole, vyhledá a zobrazí detailní informace o předmětu na základě inventárního čísla. Protože je uloženo v řetězci na první pozici, stejně jako u již používaných štítků pro evidenci školního majetku, bude moci aplikace načíst detail o předmětu i přečtením QR kódu z již zavedených štítků.

Údaje o předmětu mohou být v kódu uloženy jako alfanumerické znaky nebo text kódovaný v bajtech. Alfanumerické znaky by umožnily při stejné délce řetězce úsporu kapacity. Tím by mohl mít QR kód se stejným textem a při stejné šířce symbolu větší velikost modulů, což by znamenalo snadnější čtení kódu kamerou. Mezi podporovanými alfanumerickými znaky ale není středník, takže by se vytratila již zmíněná kompatibilita

Obrázek 5: Ukázka štítku pro evidenci školního majetku

s již používanými štítky pro evidenci školního majetku. Textový řetězec v QR kódu mého systému je tedy uložen jako text kódovaný v bajtech.

Na obrázku 6 jsou QR kódy s uloženým textovým řetězcem „1234567890;PorKr-Pol207;“. Tento text představuje příklad uložených parametrů k přístroji. Řetězec je kratší kvůli absenci dlouhého parametru v podobě názvu přístroje a menšímu množství znaků pro oddělení parametrů. Díky tomu má QR kód pouze 25 x 25 modulů a to se schopností korekce chyby 7% (ve verzi QR kódu 25-L) i se schopností korekce 15% (ve verzi 25-M). Symboly QR kódů tedy budou při stejné šířce modulu v milimetrech menší, než na již používaných štítcích pro evidenci majetku, což umožní umístění štítku i na menší předměty.



Obrázek 6: QR kód verze 25-L (vlevo) a 25-M s inventárním číslem a názvem budovy

2.4 Knihovna ZXing

Pro dekódování QR kódů je zapotřebí zpracovat obraz, ve kterém se vyskytuje kód. K tomu není nutné vyvíjet vlastní algoritmus se zpracováním obrazu, ale je možné použít již existující knihovnu. Při hledání na internetu jsem se zaměřil na open sourcové projekty. Nejvhodnější se nakonec ukázala knihovna vytvořená v rámci projektu nazvaném ZXing (čte se jako „Zebra Crossing“).

Obrázek 7: Logo projektu ZXing

Jedná se o open sourcovou knihovnu pro zpracování obrazu jedno- i dvourozměrných kódů různých formátů [4]. Ve výčtu podporovaných formátů je i QR kód, neuvádí se však podporovaný druh. Jednoduchým pokusem o dekódování dat z obrázků s různými druhy kódů online dekóderem projektu ZXing jsem zjistil podporovanou verzi. Dekódovat se nepodařilo Micro QR kód ani QR Code Model 1, ale pouze verze QR Code Model 2. Vzhledem k doporučení v ISO normě o použití Modelu 1 pouze pro uzavřené systémy jsem se možnostem knihovny u tohoto druhu ani nedivil. Co se týče Micro QR kódu jsem byl ale překvapen a proto jsem zkusil dohledat informace ve spojení s touto knihovnou a tímto druhem kódu. Na stránkách projektu jsem nakonec našel zmínku o Micro QR kódech v sekci problémů. Knihovna tento druh kódů skutečně nepodporuje a autoři dopracování podpory neplánují. Samozřejmě se ale nebrání, pokud kdokoliv podporu této verze v rámci projektu dodělá [5]. Knihovna nedokázala také dekódovat zrcadlově převrácený QR kód model 2 ani kód s invertovanými barvami (bílý vzor na černém podkladu), což by mělo odpovídat verzi 2005. Již v první části této kapitoly ale vysvětluji, že v systému jsem se rozhodl použít QR Code Model 2, protože jeho rozšířená specifikace na verzi 2005 by nebyla používána. Takže použití knihovny ZXing považuji za optimální.

Původní kód knihovny je napsán v jazyce Java s portováním na jiné jazyky. Jedním z nich je i C#, který je používán pro vývoj Silverlight aplikací, takže jejímu použití v mé aplikaci pro čtení kódů nic nebrání. ZXing knihovna je často probírána v diskuzních fórech na internetu, takže během vývoje aplikace využívající tuto knihovnu jsem se nemusel obávat, že v případě výskytu problému budu dlouho hledat řešení.

Kód knihovny je rozdělen do několika tříd, které zajišťují zpracování obrazu a dekódování různých formátů kódu. Pro použití v mé aplikaci mi stačilo použít jen třídy pro práci s QR kódy.

2.5 Příklad použití ZXing knihovny

Pro ukázku použití této knihovny uvádím kód jednoduché WindowsForms aplikace s jedním tlačítkem. Po klinutí na toto tlačítko se aplikace pokusí přečíst kód z fotografie. Ukázka je napsána v jazyce C#.

Do nově vytvořené WindowsForms aplikace jsem přidal referenci na zkompilovanou ZXing knihovnu. Na formulář aplikace jsem umístil tlačítko a k němu vytvořil událost `button1_Click`, která se vyvolá při kliknutí na něj. Celý kód události je vidět v ukázce 1. Kód funguje následovně:

1. do proměnné `Bitmap myBitmap` se načte obrázek s QR kódem
2. obrázek `myBitmap` se předá jako parametr konstruktoru třídy `RGBLuminanceSource` společně s jeho šířkou a výškou
3. ve třídě `RGBLuminanceSource` je obrázek uložen do pole typu `sbyte` a instance této třídy se předá jako parametr kontruktoru třídy `HybridBinarizer`
4. ve třídě `HybridBinarizer` dojde k výpočtu hodnot pro prahování obrazu, který se potom může převést na binární bitmapu zadáním parametru konstruktoru třídy `BinaryBitmap`
5. binární bitmapu přijímá třída `QRCodeReader` jako parametr metody `decode`, která vrací proměnnou typu `Result`; v této proměnné jsou po přečtení kromě samotného dekódovaného obsahu například i typ dekódovaného obrázkového kódu
6. dekódovaný text se vypíše do dialogového okna; pokud se dekódování nezdaří, je do dialogového okna vypsán text výjimky

```
private void button1_Click(object sender, EventArgs e)
{
    Bitmap myBitmap = new Bitmap("F:/qr.png");
    RGBLuminanceSource luminiance = new RGBLuminanceSource(myBitmap, myBitmap.Width,
        myBitmap.Height);
    HybridBinarizer binarizer = new HybridBinarizer(luminiance);
    BinaryBitmap binBitmap = new BinaryBitmap(binarizer);
    QRCodeReader qrRead = new QRCodeReader();
    Result result;
    try
    {
        result = qrRead.decode(binBitmap);
        MessageBox.Show(result.Text);
    }
        catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
```

Výpis 1: Ukázka použití ZXing knihovny

3 Aplikace pro čtení QR

3.1 Microsoft Silverlight

Silverlight je platforma pro tvorbu internetových aplikací s bohatým obsahem, které se v angličtině označují zkratkou RIA. Ty mohou být distribuovány přes web. Aplikace může běžet v prohlížeči jako součást webové stránky nebo v režimu mimo prohlížeč, který se označuje anglickým termínem out-of-browser. Od verze s číslem čtyři je Silverlight aplikace schopna používat webovou kameru zařízení, na kterém je spuštěna. Aplikace na této platformě mohou být po nainstalování pluginu spuštěny jak na PC s operačním systémem Windows a Linux, tak na strojích Mac firmy Apple [6]. Pro Windows a Mac je možné stáhnout plugin ze stránek firmy Microsoft. U operačního systému Linux je pro běh aplikace možné stáhnout plugin Moonlight, což je open sourcová implementace Silverlightu určena primárně pro Unix operační systémy [7].

Je založen na .NET Frameworku a pro vývoj lze tím pádem použít jazyky Visual Basic nebo C#. Grafické rozhraní aplikace je definováno v souboru XAML (Extensible Application Markup Language). Vývoj je možný v přostředí Visual Studio s nainstalovaným Silverlight SDK. Na této platformě jsou také založeny mobilní aplikace pro Windows Phone 7 a vyšší [6]. Vývojem aplikace pro čtení QR kódů na této platformě vzejde aplikace nezávislá na operačním systému, kterou je možné používat i bez instalace.

3.2 Požadavky na aplikaci a návrh grafického rozhraní

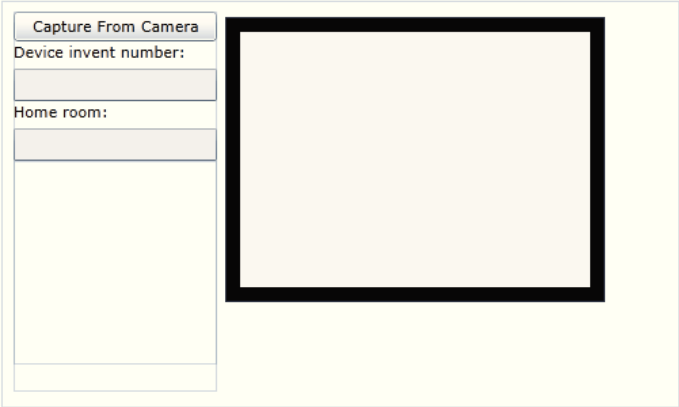
Aplikace bude v systému používána pro čtení informací z QR kódu. Proto musí mít tyto funkce:

- získání přístupu ke kameře zařízení
- zobrazení obrazu z kamery v grafickém rozhraní
- pokud je kamerou zachycen QR kód, musí aplikace provést dekódování uložených informací
- dekódované informace zobrazit v grafickém rozhraní aplikace

Navrhnuté grafické rozhraní aplikace je vidět na obrázku 8.

Grafické rozhraní je poskládáno z následujících komponent:

- tlačítek pro spuštení a zastavení snímání obrazu z kamery – zobrazeno je vždy jen jedno tlačítko, druhé je skryto
- textových polí pro zobrazení dekódovaných informací a popisků těchto textových polí
- obdelníku k zobrazení obrazu z kamery



Obrázek 8: Navržené grafické rozhraní aplikace

3.3 Popis aplikace

Aplikace se skládá z několika tříd. Jejich proměnné a metody jsou vidět na třídním diagramu v příloze na obrázku 27. Jednotlivé třídy jsou popsány v následujících odstavcích.

3.3.1 Třída CameraReader

Tato třída poskytuje připojení ke kameře zařízení, na kterém je aplikace spuštěna. Zároveň má na starosti ukládání zachyceného obrazu do proměnné, která je potom použita pro dekóvání v jiné třídě. Třída samotná obsahuje následující proměnné:

- `captureSource` – proměnná typu `System.Windows.Media.CaptureSource`; poskytuje metody pro zachycení obrazu z kamery zařízení
- `rectCameraPreview` – proměnná typu `System.Windows.Shapes.Rectangle`; pro vykreslení obdelníku s výplní; v případě této aplikace bude výplní obraz z kamery
- `bitmapForDecoding` – proměnná pro uložení zachyceného obrazu z kamery; typu `System.Windows.Media.Imaging.WriteableBitmap`
- `videoBrush` – proměnná `typu System.Windows.Media.VideoBrush`; vyplní oblast obdelníku video obsahem

S proměnnými této třídy lze pracovat pomocí těchto veřejných metod:

- `VideoBrush prepCaptureSource()` – připojí se ke kameře zařízení a obraz vrátí jako `VideoBrush`
- `VideoBrush stopCaptureSource()` – odpojí kameru zařízení a vrátí obraz s hodnotou `null`

- `void setRectangle(Rectangle fromRectangle)` – jako parametr příjmá obdelník a ten nastaví jako referenci proměnné `rectCameraPreview`
- `WriteableBitmap getBitmap()` – vrátí proměnnou této třídy `bitmapForDecoding`
- `void webcam_GrabImage()` – událostí `captureSource_CaptureImageCompleted` asynchronně zachytí obraz z kamery a uloží jej do proměnné `bitmapForDecoding`

3.3.2 Třída ZXingDecoder

Hlavní účel této třídy je dekódování obrazu zachyceného kamerou. Jsou v ní následující neveřejné proměnné:

- `isReadingSuccessfull` – proměnná typu `bool`; rozlišuje, zda se v obraze nacházel QR kód a byl úspěšně dekódován nebo ne
- `decodedString` – proměnná typu `string`; obsahuje text dekódovaný z QR kódu
- `bitmapForDecoding` – z obrazu v této proměnné se metoda `read()` pokusí dekódovat obsah QR kódu; je typu `System.Windows.Media.Imaging.WriteableBitmap`;

Dále jsou ve třídě tyto veřejné metody:

- `bool getIsReadingSuccessfull()` – vrátí obsah proměnné `isReadingSuccessfull`
- `string getDecodedString()` – vrátí obsah proměnné `decodedString`
- `void setBitmapForDecoding(WriteableBitmap newBitmap)` – jako parametr přijímá nový obrázek a ten ukládá do proměnné `bitmapForDecoding`
- `void read()` – pokusí se z proměnné `bitmapForDecoding` dekódovat obsah QR kódu

Pro dekódování je využita knihovna ZXing. V ukázce 2 je zdrojový kód metody `read()`. Z ukázky je patrné, jak je dekódování touto knihovnou v aplikaci řešeno. Do proměnné `decodedString` je v případě neúspěšného dekódování ukládán text výjimky. Tuto výjimku je v aplikaci následně možné vypisovat. Finální systém ale obsahuje aplikaci, ve které je toto vypisování vypnuto, protože uživatel nepotřebuje vidět důvod toho, proč se nepodařilo z obrazu kamery dekódovat obsah QR kódu.

```
public void read()
{
    isReadingSuccessfull = false;
    QRCodeReader qrRead = new QRCodeReader();
    RGBLuminanceSource luminiance = new RGBLuminanceSource(this.bitmapForDecoding,
        this.bitmapForDecoding.PixelWidth, this.bitmapForDecoding.PixelHeight);
    HybridBinarizer binarizer = new HybridBinarizer(luminiance);
    BinaryBitmap binBitmap = new BinaryBitmap(binarizer);
    Result result;
    try
```

```
        {
            result  = qrRead.decode(binBitmap);
            this.decodedString = result.Text;
            this.isReadingSuccessfull = true;
        }
            catch (Exception ex)
        {
            this.decodedString = ex.ToString();
        }
    }
```

Výpis 2: Metoda read()

3.3.3 Třída ReadedDevice

Tato třída představuje přečtené zařízení. V QR kódu je uloženo inventární číslo a domovská místnost přístroje ve formě textu a tyto parametry jsou odděleny středníkem. Metoda této třídy rozdělí podle středníku text do jednotlivých parametrů. Třída se tedy skládá z těchto veřejných proměnných:

- `deviceInventaryNumber` – proměnná typu `string`; v ní je uloženo po rozdělení řetezce inventární číslo
- `deviceHomeRoom` – proměnná typu `string`; v ní je uložena po rozdělení řetezce domovská místnost

Pro rozdělení řetězce do proměnných je ve třídě tato veřejná metoda:

- `void parseFromReadedString(string readedString)` – rozdělí parametr `readedString` podle středníků a uloží je do proměnných `deviceInventaryNumber` a `deviceHomeRoom`

3.3.4 Třída MainPage

Ve třídě `MainPage` je definováno chování uživatelského rozhraní a jsou v ní použity instance všech již popsaných tříd. Proměnné jsou tedy následující:

- `myCameraReader` – proměnná typu `CameraReader`
- `myZXingDecoder` – proměnná typu `ZXingDecoder`
- `myReadedDevice` – proměnná typu `ReadedDevice`
- `myDispatcherTimer` – proměnná typu `System.Windows.Threading.DispatcherTimer`; poskytuje časovač, který vykonává událost v samostatném vlákně

Metody ve třídě jsou neveřejné a jsou následující:

- `void prepCaptureSource()` – zavolá metodu `prepCaptureSource()` v proměnné `myCameraReader` a vyplní obsah obdelníku v grafickém rozhraní obrazem z kamery
- `void openDetail(string inventnum)` – otevře stránku zařízení, jehož QR kód byl přečten

Kromě metod jsou ve třídě tyto události:

- `void btnCaptureFromCamera_Click()` – zavolá metodu `prepCaptureSource()` v této třídě
- `void btnStopCapturing_Click()` – zastaví časovač `myDispatcherTimer()` a zavolá metodu `stopCaptureSource()` v proměnné `myCameraReader`
- `void Each_Tick()` – událost pro zachycení obrazu z kamery metodou `webcam_GrabImage()` a dekódování textu z tohoto obrazu metodou `read()`

V konstruktoru třídy se vytvoří instance všech proměnných a do časovače `myDispatcherTimer` se přidá událost `Each_Tick`. Interval časovače se nastaví na 100 milisekund. V ukázce 3 je vidět kód události `Each_Tick()`. Nejprve se v ní zachytí obraz z kamery, potom se zachycený obraz předá třídě pro dekódování a ta se pokusí v obrazu najít QR kód a přečíst jeho obsah. Pokud bylo přečtení úspěšné, uloží se do `myReadedDevice` obsah QR kódu. Následně se do textových polí vypíší informace o přečteném zařízení a zavolá metoda `openDetail`, které je jako parametr předáno inventární číslo právě přečteného zařízení.

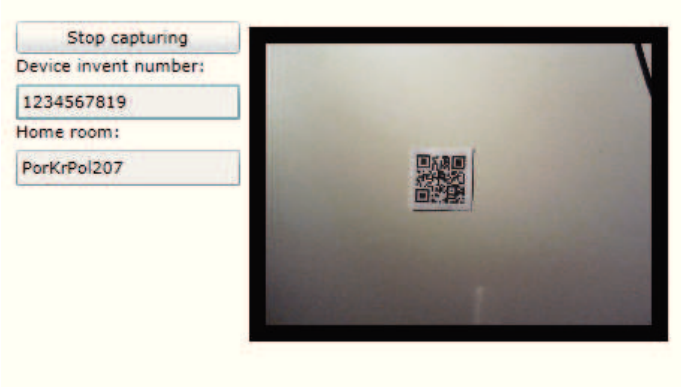
```
public void Each_Tick(object o, EventArgs sender)
{
    myCameraReader.webcam_GrabImage();
    myZXingDecoder.setBitmapForDecoding(myCameraReader.getBitmap());
    myZXingDecoder.read();
    if (myZXingDecoder.getIsReadingSuccessfull() == true)
    {
        myReadedDevice = new ReadedDevice();
        myReadedDevice.parseFromReadedString(myZXingDecoder.getDecodedString());
        tbDevice.Text = myReadedDevice.deviceInventaryNumber;
        tbRoom.Text = myReadedDevice.deviceHomeRoom;
        openDetail(myReadedDevice.deviceInventaryNumber);
    }
}
```

Výpis 3: Událost Each_Tick()

3.4 Testování finální aplikace

Pro testování aplikace jsem využil externí webovou kameru připojenou do USB portu počítače. Při reálném použití aplikace předpokládám, že bude častěji využívána kamera integrovaná v zařízení, protože externí webovou kameru u sebe nikdo běžně nenosí. Čtení externí kamerou je ale pohodlnější a dost připomíná markování čárových kódů na

zboží v obchodě. Při inventůře si dokážu představit, že bude aplikace využívána právě s použitím externí kamery. Moje externí kamera má navíc funkci automatického ostření, takže dokázala přečíst QR kód ve verzi 25-M o velikosti 2 x 2 cm jak ze vzdálenosti 20 cm (obrázek 9), tak ze vzdálenosti 2 cm (to je zachyceno na obrázku 10).



Obrázek 9: Aplikace při přečtení kódu ze vzdálenosti 20 cm

Obrázek 10: Aplikace při přečtení kódu ze vzdálenosti 2 cm

Výsledná Silverlight aplikace umožňuje číst jen QR kódy model 2, což plyne ze schopností knihovny ZXing. Aplikace bude použita pro čtení kódů, které bude generovat mnou vytvořený systém a ty budou generovány stejnou knihovnou. Toto omezení tedy hodnotu vytvořené aplikace nijak nesnižuje.

4 Databáze pro systém

Data pro informační systémy je výhodné ukládat do databáze. Pro manipulaci s daty je zapotřebí určitých softwarových prostředků, které se nazývají systém řízení báze dat (SŘDB) [8]. Firma Microsoft poskytuje své řešení SŘDB pod názvem Microsoft SQL Server a to hned v několika edicích. V systému používám edice Express, která je zdarma. Jedním z omezení této edice je maximální kapacita 10 GB uložených dat v databázi [9]. Pokud by systém tuto kapacitu začal v budoucnu naplňovat, bude potřeba přejít na jinou edice databáze s vyšší kapacitou.

4.1 Návrh databáze

Podle požadavků na systém je potřeba u přístrojů evidovat následující údaje:

- inventární číslo
- název přístroje
- název přístroje v původním jazyce
- výrobce přístroje
- webové stránky výrobce
- místnost, do které přístroj patří
- místnost, ve které se přístroj aktuálně nachází
- fotografie přístroje
- soubory přístroje

Hlavní tabulkou databáze bude tabulka s přístroji. Kdybych ale vytvořil jednu tabulku s výše uvedenými parametry, bylo by v tabulce velké množství redundantních dat. Více přístrojů může mít stejného výrobce a místnost, takže bude výhodné vytvořit číselník výrobců a místností. Dále může být každé přemístění přístroje uloženo jako samostatný záznam s časovým údajem označujícím jeho platnost. Záznamy o aktuálním umístění přístroje mohou také využívat číselník místností. Na grafickém znázornění výsledného návrhu databáze 28 je vidět použití těchto číselníků, relace mezi záznamy v tabulkách a jejich kardinalita. Z ní plyne, že jeden záznam výrobce může patřit k více záznamům přístroje a jeden záznam místnosti může patřit k více záznamům přístroje a více záznamům o poloze přístroje. Dále pak, že k jednomu záznamu o přístroji může existovat více záznamů o jeho poloze, více záznamů s fotografiemi a také více záznamů se soubory. V návrhu jsou také vidět všechny atributy tabulek. Popis atributů a jejich datové typy jsou uvedeny v příloze v tabulkách 3, 4, 5, 6, 7 a 8.

Společné atributy všech šesti databázových tabulek jsou uvedeny v tabulce 1. Jak je z ní patrné, u každého záznamu bude v systému evidováno, kdo a kdy jej vytvořil a také

název atributu	**datový typ**	**popis**
InsertedDate	datetime	datum a čas vložení záznamu
InsertedByUserName	nvarchar(256)	uživatel, který vložil záznam
UpdatedDate	datetime	datum a čas poslední aktualizace záznamu
UpdatedByUserName	nvarchar(256)	uživatel, který naposledy změnil záznam
Deleted	bit	1 = neplatný záznam, 0 = platný záznam

Tabulka 1: Společné atributy všech tabulek

kdo a kdy jej naposledy změnil. Kromě toho nebudou záznamy při mazání v systému odstraňovány z databáze, ale pouze zneplatňovány. To bude užitečné na příklad v případě nechtěného smazání přístroje v systému uživatelem, protože bude možné informace o záznamu vyhledat přes administrátora systému.

4.2 Realizace databáze

Po návrhu databáze jsem mohl přejít k jejímu vytvoření. To lze provést v programu Microsoft SQL Server Management Studio. Databáze a tabulky se tímto softwarem mohou vytvořit přes designer na lokálním počítači, potom se odpojí databázový soubor MDF obsahující data a LDF soubor s logy databáze a tyto soubory potom přesunout na server, na kterém bude databáze pro systém provozována. Tam se databázové soubory připojí k běžícímu Microsoft SQL Serveru nebo umístí do adresáře, ze kterého aplikace soubory použije při provádění dotazu. Druhou možností je napsat skript v jazyce T-SQL, který vytvoří databázi i s tabulkami a tento následně spustit na serveru s provozovaným Microsoft SQL Serverem. Já jsem zvolil první variantu a databázi vytvořil designerem. V Management Studio je i pro již vytvořenou databázi možné nechat vygenerovat skript, který je ji schopný vytvořit včetně tabulek a relací. Tento skript jsem si po vygenerování uložil pro případ, že by bylo potřeba vytvořit novou a prázdnou databázi. Při vývoji webové aplikace pro přístup k databázi jsem skript několikrát použil, protože jsem občas potřeboval začít s prázdnou databází.

4.3 LINQ a repository třída

V informačních systémech a aplikacích je možné pro přístup k datům používat nástroje pro objektově relační mapování. Jedním z nástrojů, který k tomu slouží je LINQ to SQL od Microsoftu. Pomocí tohoto nástroje je možné vytvořit třídy mapované na relační objekty. Ve Visual Studiu jsem do projektu s webovým rozhraním systému přidal dbml soubor, ve kterém jsem nastavil mapování na tabulky databáze. Visual Studio po tomto kroku automaticky vygeneruje každé tabulce odpovídající třídu s vlastnostmi odpovídajícím atributům tabulky. Pokud je v databázi nastavena relace mezi záznamy tabulek, vytvoří se v dbml souboru asociace mezi třídami, které jsou na tyto tabulky mapovány. Třídní diagram dbml souboru je vidět na obrázku 29 v příloze. Podobnost mezi vytvořenými třídami a skrukturou tabulek v databázi je zřejmá.

Ve webovém rozhraní jsem pro přístup k datům vytvořil třídu `InventRepository`. Webové rozhraní tedy čerpá z výhod návrhového vzoru repository [10]. Díky němu se neřeší přístup k datům v každé třídě zvlášť, ale použije se jedna třída, ve které jsou napsány metody pro získání dat ve všech podobách, v jakých je aplikace potřebuje. Tím je v systému oddělena vrstva pro získání dat od doménového modelu. V ukázkovém výpise zdrojového kódu 4 uvádím dvě metody v repository třídě `FindAllDevices()` a `GetDevice(int id)`. V nich je vidět použití LINQ syntaxe pro získání dat a podmínka pro výběr (protože záznamy se z databáze neodstraňují, ale pouze se zneplatní parametrem `deleted`).

```
public IQueryable<Device> FindAllDevices()
{
    return from device in db.Devices
        where device.Deleted == false
        orderby device.DeviceInventaryNum
        select device;
}

public Device GetDevice(int id)
{
    return db.Devices.SingleOrDefault(d => d.DeviceId == id && d.Deleted == false);
}
```

Výpis 4: Ukázka kódu repository třídy

Výhodou řešení přístupu k datům přes návrhový vzor repository je také to, že v případě potřeby výměny databáze Microsoft SQL za jinou stačí přepsat repository třídu. Není nutné procházet celý kód aplikace a přepisovat jej všude tam, kde se získávají data.

5 Webové rozhraní pro přístup k databázi

5.1 ASP.NET MVC

ASP.NET MVC je technologie pro vývoj webových aplikací používající platformu .NET firmy Microsoft. Byla představena v roce 2007 jako reakce na kritiku ASP.NET Web Forms. ASP.NET MVC využívá návrhový vzor MVC (Model-View-Controller), jehož vznik je datován do roku 1978 na projektu Smalltalk [11].

Dnes je tento návrhový vzor populární u webových aplikací. To má dva hlavní důvody. Prvním z nich je přirozený průchod všemi komponentami tohoto vzoru při jejím ovládání. Obvykle klikneme na stránce aplikace na nějaký odkaz a vyvolá se akce (ta se nachází v kontroléru). Akce provede něco s daty (ty jsou v modelu) a následně zobrazí výsledek zpátky na stránce (použije se komponenta View, což je šablona pro webovou stránku, ve které se dynamicky mění obsah). Druhým důvodem je to, že webové aplikace téměř vždy využívají několik technologií rozdělených do samostatných vrstev. Například databázi, dynamický kód a vlastní stránky v html. A přirozenou kombinaci těchto technologií právě MVC vzor umožňuje.

Pro realizaci systému v ASP.NET MVC jsem se rozhodl z důvodu výše uvedených výhod návrhového vzoru MVC. Dalším důvodem pak bylo vývojové prostředí, framework a programovací jazyk. Vývoj Silverlight aplikace pro čtení QR kódů probíhal ve Visual Studiu v .NET Frameworku a jazyce C#. A protože ASP.NET MVC webovou aplikaci je možné vyvinout ve stejném prostředí se stejným frameworkem a kód psát ve stejném jazyce, rozhodl jsem se právě pro něj.

5.2 Požadavky na webovou aplikaci

Aplikace pro administraci a prohlížení obsahu databáze bude v případě řešení přes webovou aplikaci snadno přístupná přes internet a její použití bude platformě nezávislé. Existuje několik požadavků, které musí aplikace splnit.

1. v aplikaci by měl být zobrazen obsah databáze a evidované přístroje je potřeba mít možnost vyhledávat; pro vybrané uživatele musí být k dispozici vytváření, editace a mazání přístrojů
2. všichni uživatelé nesmí mít možnost přidávat, měnit a mazat záznamy
3. z důvodu zabezpečení přístupu k datům je třeba v aplikaci požadovat přihlášení přes uživatelské jméno a heslo
4. již smazané položky by měly být přístupné pro čtení
5. v aplikaci by měla být možnost hromadného nahrání přístrojů

Každému bodu požadavků je věnována samostatná část této kapitoly, ve které je popsán způsob jeho naplnění.

5.3 CRUD

Touto zkratkou se označují funkce pro vytváření (Create), čtení (Read), aktualizaci (Update) a mazání (Delete) záznamů v systémech [12]. ASP.NET MVC používá pro ovládání aplikace kontroléry, což jsou třídy dědící z abstraktní třídy Controller. K vytváření, čtení, aktualizaci a mazání musí proto v kontrolérech existovat příslušné akční metody. Výsledek po provedení akční metody se zobrazí pomocí View šablony.

5.3.1 ProducerController, RoomController a jejich View šablony

V těchto kontrolérech jsou uloženy akční metody pro CRUD číselníku výrobců a místností. Oba kontroléry jsou téměř shodné a liší se jen použitým modelem. Akční metody jsou následující:

- `public ActionResult Index()` – pro zobrazení seznamu výrobců nebo místností
- `public ActionResult Edit(int id)` – metoda pro zobrazení stránky pro editaci
- `public ActionResult Edit(int id, FormCollection formValues)` – POST metoda pro editaci, která příjmá údaje vyplněné na stránce a změny ukládá do databáze
- `public ActionResult Create()` – metoda pro zobrazení stránky pro vytvoření nového výrobce nebo místnosti
- `public ActionResult Create(Room room)` – POST metoda pro přidání, která příjmá nově vytvořenou instanci objektu (v tomto případě místnoti) a ukládá ji do databáze
- `public ActionResult Delete(int id, FormCollection formValues)` – POST metoda pro smazání položky v systému; metoda vrací jen `True` nebo `False` a je volána na stránce javascriptem.

View šablony pro zobrazení výsledku mají stejný název, jako uvedené metody. Výjimku tvoří metoda `Delete`, která vlastní View šablonu nemá. Zavolání této metody se potvrzuje v dialogovém okně a po potvrzení a provedení zůstává uživatel na stránce Index.

5.3.2 LocationController a jeho View šablony

Pozice přístrojů mají také své akční metody a šablony pro vzhled stránek. Ty se nachází v tomto kontroléru. Od akčních metod pro výrobce a místnosti se liší v přidání atributu metodám `Index(int id)` a `Create(int id)`. Tímto parametrem je id záznamu přístroje, pro který se má zobrazit seznam pozic nebo vytvořit nová pozice. Druhý rozdíl je v potřebě předat View stránce `Edit` nebo `Create` aktuální seznam místností z číselníku. Tyto stránky jej potřebují pro zobrazení rozbalovací nabídky pro výběr místnosti,

ve které se přístroj nachází. Seznam je možné předat použitím třídy „ViewModel“, což je třída použitá jako model pro zobrazované View. Tuto třídu jsem vytvořil a pojmenoval ji `LocationFormViewModel`. Obsahuje tyto proměnné:

- `DeviceLocation DeviceLocation` – pozice přístroje, která se má ve View zobrazit
- `SelectList Rooms` – rozbalovací nabídka místností; na stránce se pozici vybere místnost, ve které se přístroj nachází

Instance této třídy se vytvoří v akčních metodách a také se v nich přiřadí hodnoty proměnným této třídy. Ve View `Edit` nebo `Create` se potom do příslušných textových polí přiřadí hodnoty parametrů proměnné `DeviceLocation` a do rozbalovací nabídky se nahraje `SelectList Rooms`. Toto je vidět v ukázce kódu číslo 5.

```
<div class="editor-label">Select room</div>
<div class="editor-field">
    <%= Html.DropDownListFor(model => model.DeviceLocation.RoomId, Model.Rooms ) %>
</div>

<div class="editor-label">Location is valid from:</div>
<div class="editor-field">
    <%= Html.TextBoxFor(model => model.DeviceLocation.DeviceLocationValidFrom) %>
    <%= Html.ValidationMessageFor(model => model.DeviceLocation.DeviceLocationValidFrom)
        %>
</div>

<div class="editor-label">Location is valid to:</div>
<div class="editor-field">
    <%= Html.TextBoxFor(model => model.DeviceLocation.DeviceLocationValidTo)%>
    <%= Html.ValidationMessageFor(model => model.DeviceLocation.DeviceLocationValidTo)%>
</div>
```

Výpis 5: Naplnění hodnot textových polí a rozbalovací nabídky z ViewModelu

5.3.3 DeviceController a jeho View šablony

Celý systém je především o inventarizaci přístrojů a jejich CRUD je definován metodami, které jsou uloženy v kontroléru DeviceController. Index stránka přístrojů obsahuje oproti index stránkám výrobců a místností navíc filtrování výsledného seznamu a také jeho stránkování. To je umožněno použitím „ViewModelu“. V případě Index stránky přístrojů je modelem třída `DeviceIndexViewModel` a skládá se z těchto proměnných:

- `IEnumerable<Device> Device` – seznam přístrojů, které se mají ve View zobrazit
- `SelectList Producers` – rozbalovací nabídka výrobců pro filtr
- `SelectList Rooms` – rozbalovací nabídka místností pro filtr

- `SelectList PageSizes` – rozbalovací nabídka pro zvolení počtu položek na stránku
- `int page_count` – na kolik stránek se seznam přístrojů vejde
- `int page_size` – zvolený počet položek na stránku
- `int page` – identifikuje aktuální stránku
- `string inv_number` – pokud není prázdná, je seznam přístrojů filtrován podle inventárního čísla obsahující hodnotu této proměnné
- `string Producer` – pokud není prázdná, je seznam přístrojů filtrován podle výrobců, jejichž název je roven této proměnné
- `string Room` – pokud není prázdná, je seznam přístrojů filtrován podle místností, jejichž název je roven této proměnné
- `string dev_name` – pokud není prázdná, je seznam přístrojů filtrován podle názvu obsahující hodnotu této proměnné
- `string orig_name` – pokud není prázdná, je seznam přístrojů filtrován podle původního názvu obsahující hodnotu této proměnné

Hodnoty proměnných ViewModelu se přiřazují v akční metodě `Index`. Tato metoda přijímá argumenty `string inv_number`, `string producer`, `string room`, `string dev_name`, `string orig_name`, `int page` a `int page_size`. Argumenty jsou metodě předávány přes url.

View `Details` (zobrazení detailu přístroje) a `DeviceForm` (zobrazuje se při vytváření a editaci přístroje), používají také ViewModel. Ten jsem pojmenoval `DeviceFormViewModel` a obsahuje tyto proměnné:

- `Device Device` – přístroj, který se má ve View zobrazit
- `DeviceLocation ActualDevLocation` – aktuální pozice přístroje
- `SelectList Producers` – rozbalovací nabídka výrobců; na stránce se přístroji vybere výrobce, který u něj má být veden
- `SelectList Rooms` – rozbalovací nabídka místností; na stránce se přístroji vybere místnost, která u něj má být vedena jako domovská

Hodnoty se proměnným přiřazují v akčních metodách daného View. Proměnná `ActualDevLocation` se používá pro View `Details`. Proměnné `Producers` a `Rooms` se používají pro View `DeviceForm`.

5.3.4 ImageController, FileController a jejich View šablony

Kontroléry fotografií a souborů přístroje jsou skoro shodné. Oba musí řešit nahrání souboru při POSTu. Ve View pro nahrání souboru se přes tlačítko „Procházet“ vybere soubor a po kliknutí na `Save` se soubor nahraje. Nahrání je řešeno zpracováním argumentu `HttpPostedFileBase file` v POST metodách `Edit` a `Create` obou kontrolérů.

Fotografie ani soubory přístrojů nemají View šablony a akční metody pro Index stránky, na kterých by se zobrazoval jejich seznam. Tyto stránky pro ně nejsou zapotřebí, protože fotografie i soubory jsou zobrazovány v detailu přístroje.

5.3.5 Ostatní View šablony webové aplikace

Kromě View šablon pro určitou třídu datového modelu exituje šablona typu MasterPage. V ní se definuje obsah společný pro všechny stránky celé webové aplikace. V mém webovém rozhraní je jedna tato šablona a jmenuje se `Site.Master`. Obsahuje deklaraci výsledného html dokumentu, obsah mezi tagy `<head>` (reference na soubory se styly a javascript knihovny) a také jsem do ní uložil javascriptový kód. Ten je volán při:

- odstraňování libovolného záznamu – zobrazení dialogového okna pro potvrzení smazání a pro odeslání POST požadavku bez načítání celé stránky
- odemknutí uživatelského účtu a generování nového hesla administrátorem – pro odeslání POST požadavku bez načítání celé stránky
- zobrazení dialogového okna s aplikací pro čtení QR kódů – blíže popsáno v podkapitole „Integrace aplikace pro čtení QR kódu do webové aplikace“
- zobrazení kalendáře jQuery knihovny – pro nastavení hodnoty data a času u textových polí, která jsou pro proměnné typu `DateTime`

Díky uložení javascriptových kódů na stránku `Site.Master` nemusí být uloženy v každém View zvlášť.

Na této stránce je také uložen odkaz pro přihlášení uživatele a nabídka pro navigaci mezi přístroji, číselníkem výrobců, číselníkem místností a dalšími stránkami.

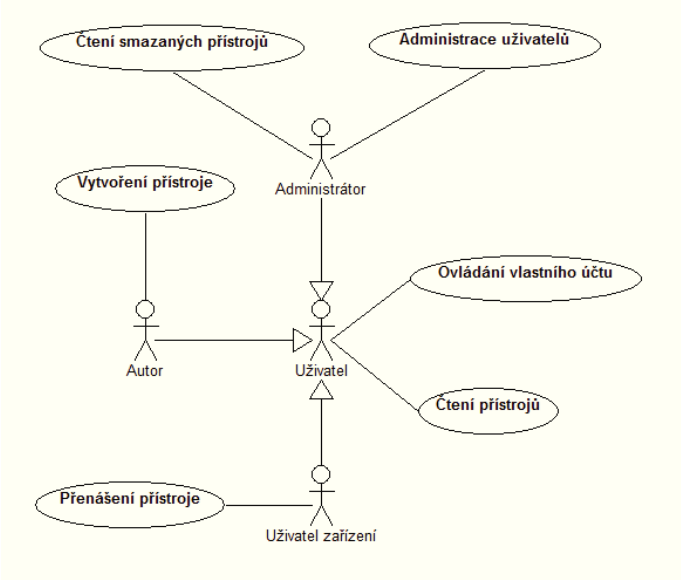
5.4 Vytvoření rolí pro webovou aplikaci

Pro splnění druhého bodu požadavků je ideálním řešením vytvoření několika rolí s určitými právy a ty následně přidělovat uživatelům. Use-case diagram na obrázku 11 zobrazuje případy užití aplikace jednotlivými rolemi. Je v něm znázorněna generalizace role Uživatel. Z té vyplývá, že práva k operacím role Uživatel mají i všechny tři další role.

5.4.1 Role Uživatel

Každý přihlášený uživatel může v systému provádět tyto akce:

- přihlášení



Obrázek 11: Use Case diagram webové aplikace

- odhlášení
- změnu hesla svého účtu
- vyhledání přístroje
- zobrazení seznamu přístrojů, výrobců a místností
- zobrazení detailu přístroje, jeho fotografie, soubory a všechny pozice

Tato role tedy není přímo vedena jako záznam mezi ostatními rolemi. Pro získání jejich oprávnění bude pouze potřeba, aby byl uživatel ve webovém rozhraní přihlášen.

5.4.2 Role Uživatel přístrojů

Uživatel v této roli má práva na tyto operace:

- vytvoření nové pozice přístroje
- editace existující pozice přístroje
- smazání pozice přístroje

5.4.3 Role Autor

V roli Autor může uživatel ve webové aplikaci:

- vytvořit, editovat a mazat přístroje, jejich soubory a fotografie
- importovat přístroje do databáze systému z csv souboru
- vytvořit, editovat a mazat výrobce
- vytvořit, editovat a mazat místnosti

5.4.4 Role Administrátor

Tato role poskytuje uživateli práva na:

- editaci a smazání uživatelů
- generování nového hesla uživatelům
- čtení smazaných přístrojů, jejich fotografií, souborů, výrobců a místností

5.5 Autentizace a autorizace

Pro zabezpečení přístupu je ve webové aplikaci použit systém autentizace a autorizace. Při vytvoření nového projektu ASP.NET MVC přes Visual Studio je vytvořena ukázková webová aplikace, která již obsahuje registraci uživatelů, přihlášení a změnu hesla. Součástí aplikace jsou komponenty model „AccountModels“, kontroler „AccountControler“ a složka s View „Account“. Projekt má tedy již vytvořeny modely pro autorizaci a autentizaci, akční metody pro jejich použití a šablony zobrazovaných stránek, které jsou pro použití potřeba. AccountModel používá abstraktní třídu System.Web.Security.MembershipProvider, která slouží pro autentizaci i autorizaci. V souboru `web.config` se konfiguruje, která databáze poskytuje data pro tento model. Výchozím nastavením je databázový soubor ASPNETDB.MDF v adresáři App_data. Ukázková webová aplikace si s vytvořením prvního uživatele tuto databázi sama vytvoří.

Ve své webové aplikaci jsem použil toto hotové řešení, protože obsahuje také metody pro vytvoření šifrovaného cookie souboru, který si pamatuje přihlášení uživatele v prohlížeči a další užitečné metody. Použití již hotového nástroje od Microsoftu je jednodušší a taky bezpečnější, protože při vytváření vlastního způsobu autorizace a autentizaci nemusí programátor zohlednit všechny bezpečnostní požadavky. Pro můj projekt používám z vytvořené databáze pro AccountModel tabulky aspnet_Applications, aspnet_Users, aspnet_Membership, aspnet_Roles a aspnet_UsersInRoles.

Do tabulky aspnet_Applications se při registraci vloží záznam pro aplikaci „\“. Jedná se o hlavní aplikaci MVC projektu a tento záznam používají další tabulky. V tabulce aspnet_Users jsou uložena uživatelská jména pro přihlášení. Hesla k uživatelským jménům do aplikací (v případě mého systému do jedné aplikace) jsou v tabulce aspnet_Membership. Heslo v tabulce je hashované, takže po jeho uložení není možné získat jeho textovou podobu. Vytvořenými metodami je možné pouze zjistit, zda text hesla při přihlášení odpovídá heslu uloženému v tabulce. Tím je zaručeno, že ani administrátor s přístupem k záznamům v databázi nemůže zjistit uživatelské heslo a zneužít tak uživateli účet bez

jeho vědomí. Posledními použitými tabulkami jsou aspnet_Roles a aspnet_UsersInRoles. V tabulce aspnet_Roles jsou uloženy role do aplikací, jejich název a popis. Pro systém budou potřeba tři role a proto jsem do této databázové tabulky vložil záznamy uvedené v tabulce číslo 2 s vyplněným ApplicationId stejnou hodnotou, kterou má záznam aplikace „\“ v tabulce aspnet_Applications. Protože uživatel může mít víc než jednu roli a jednu roli může mít víc než jeden uživatel, je použita také pomocná tabulka aspnet_UsersInRoles. Do ní jsem vložil první záznam pro uživatele, kterého jsem při vývoji potřeboval v roli Administrátor. Po vložení záznamu a přihlášení přes tohoto uživatele bude možné přidělovat práva dalším uživatelům.

RoleName	LoweredRoleName
admin	admin
author	author
device user	device user

Tabulka 2: Záznamy vložené do tabulky rolí

Aby mohly určité metody webové aplikace používat jen vybrané role, je zapotřebí to u metod definovat. První možností jak toho dosáhnout je nastavit omezení na použití všem akčním metodám v celém kontroléru. Provede se to tak, že se před jeho definici uvede potřebné omezení. V ukázce zdrojového kódu 6 uvádím nastavení, díky kterému mohou všechny akční metody v `DeviceController` používat jen přihlášení uživatelé. Kdyby bylo potřeba omezení pouze na určitou roli, uvedl by se jako parametr název role. Například pro omezení na roli „autor“ by bylo v kódu `[Authorize(Roles = "author")]`. Druhou možností je omezit jen vybrané metody kontroléru. V takovém případě se uvede omezení přímo před metodu, což je vidět v ukázce 7. Metodu `Edit` v ukázce mohou používat pouze uživatelé s rolí `device user`.

```
[Authorize]
public class DeviceController :  Controller
{
 ...
```

Výpis 6: Přístup ke kontroléru jen přihlášeným uživatelům

```
[Authorize(Roles = "device user")]
public ActionResult Edit(int  id)
{
 ...
```

Výpis 7: Přístup k metodě jen uživatelům přístrojů

Podle omezení práv jednotlivých rolí jsem tady nastavil parametry celým kontrolérům nebo jen některým metodám. Protože by ale nebylo vhodné uživatelům zobrazovat na stránce odkazy na akční metody, které stejně nemohou volat, je zapotřebí jim tyto možnosti skrýt. To jsem provedl přidáním podmínek do jednotlivých View. Jako příklad uvádím výpis 8, který je z kódu View `Details` pro stejně pojmenovanou metodu

z `DeviceController`. Touto podmínkou je zajištěno, že uživateli bez role `author` se nezobrazí odkaz pro editaci přístroje.

```
<%if(User.IsInRole("author"))
    {%>
    <%= Html.ActionImage("Edit", new { id = Model.Device.DeviceId }, "/Content/bigedit.png", "Edit␣
        device")%>
<%} %>
```

Výpis 8: Podmíněné zobrazení ve View dle role

5.6 Náhled na smazané položky

Všechny databázové tabulky systému mají parametr, který označuje platnost záznamu. Ten jsem pojmenoval `deleted`. Když uživatel ve webovém rozhraní odstraní například přístroj, pouze se změní hodnota tohoto parametru, ale záznam zůstane v databázi uložen. Tyto zneplatněné záznamy si může zobrazit uživatel v roli administrátora přes svou záložků. V této záložce je podmenu s odkazy pro zobrazení smazaných přístrojů, výrobců, místností, fotografií a souborů. Kliknutí na odkaz volá akční metodu z kontroléru `AdminController`, který je oprávněn používat pouze uživatel s rolí `admin`.

Výsledné zobrazení neplatných záznamů je podobné, jako u těch platných. Jedním z rozdílů je volání jiné metody z třídy InventRepository, aby se vrátily jen neplatné záznamy. Druhý rozdíl zobrazení je v tom, že v něm není možnost záznamy editovat. Proto v něm nejsou zobrazeny ikony s odkazy. Odstraněné položky zůstávají v systému k dispozici jen pro případ, že by jeho smazání provedl uživatel omylem a bylo by potřeba informace o položce opět získat.

5.7 Import do databáze z CSV

V systému je plánováno evidovat několik tisíc přístrojů a jiných předmětů. Jejich vkládání po jednom by bylo velmi zdlouhavé. Proto jsem připravil v systému import přístrojů ze souboru ve formátu CSV. Tento import je možné použít jak při prvotním plnění systému, tak i pro libovolné dodatečné přidání nových přístrojů do databáze systému.

Tuto funkci má k dispozici uživatel v roli autora ve webovém rozhraní, konkrétně na stránce se seznamem přístrojů. Po kliknutí na odkaz „CSV IMPORT“ se mu zobrazí stránka, kde si může stáhnout soubor s pojmenovanými sloupci. Tento soubor se generuje přímo v akční metodě (není tedy nikde v systému přímo uložen). Po vyplnění souboru jej uživatel vybere přes tlačítko „Procházet“ a potom jej může nahrát. Po nahrání se zpracuje následovně:

1. Přečte se první řádek souboru se záhlavím.
2. V cyklu se čte zbytek souboru až do konce po řádcích.
3. Každý řádek se rozdělí podle středníků do pole proměnných typu string.

4. Vytvoří se nová instance přístroje (Device) a přiřadí se jí z pole tyto atributy: inventární číslo, název, název v původním jazyce, popis.

5. Podle názvu výrobce z pole se dohledá, zda tento výrobce (Producer) již existuje – pokud ano, přiřadí se tento výrobce přístroji; pokud ne, vytvoří se nová instance výrobce s tímto názvem, ale zatím se do databáze neuloží

6. Podle názvu místnosti z pole se dohledá, zda tato místnost (Room) již existuje – pokud ano, přiřadí se tato místnost přístroji; pokud ne, vytvoří se nová instance místnosti s tímto názvem, ale zatím se do databáze neuloží

7. Zkontroluje se, zda je přístroj validní (jedna z kontrol je na to, zda nepoužívá již existující inventární číslo) – pokud je validní, provede se jeho přidání do databáze (včetně případného nového výrobce nebo místnosti) a přidá se také do seznamu úspěšně naimportovaných přístrojů; pokud není, nic se do databáze neuloží (ani případný nový výrobce nebo místnost)

8. Pokračuje se od začátku cyklu

9. Po provedení importu se zobrazí seznam úspěšně naimportovaných přístrojů

5.8 Vzhled a grafika webového rozhraní

Vzhled systému je uložen v css souboru s kaskádovými styly, což umožnilo definici vzhledu všech stránek aplikace na jednom místě. Grafika použitá na ovládací tlačítka v systému byla vytvořena softwarem Libre Office Writer Portable verze 4.0.2.2. LibreOffice je licencován pod licencí LGPLv3. To znamená, že je možné jej používat zdarma jak pro osobní, tak pro komerční účely [13]. Grafika je uložena ve vektorové podobě do souboru ODT. V budoucnu je tedy možné upravit barvy, pokud by bylo potřeba v systému změnit barevné schéma. Následně byly obrázky zmenšeny na potřebnou velikost freeware programem IrfanView a uloženy do formátu PNG. Tento program jsem také použil pro úpravu všech obrázků v této diplomové práci. IrfanView je k dispozici zdarma pro vzdělávací účely [14].

5.9 Integrace aplikace pro čtení QR kódu do webové aplikace

Jak už jsem zmiňoval ve třetí kapitole, Silverlight je platforma pro vývoj RIA aplikací, které jsou součástí internetové stránky. Rozhodl jsem se proto umístit aplikaci pro čtení QR kódů do webového rozhraní pro přístup k databázi. To umožní urychlení případných oprav parametrů evidovaného přístroje. Stačí načíst QR kód přístroje Silverlight aplikací, ta po dekódování jeho obsahu přesměruje uživatele na stránku s detailem přístroje ve webové aplikaci systému a pokud má uživatel práva pro editaci přístrojů, může si zvolit editaci přístroje přímo na této stránce.

Silverlight aplikace tedy provádí po dekódování kódu přesměrování na adresu, kterou je zavolána akční metoda `DetailsFromInventnum(string inventnum)`. Ta přijímá jako parametr inventární číslo z QR kódu. Zdrojový kód, který po přečtení provede

přesměrování je v ukázce 9. Nejprve se do proměnné `host` uloží adresa, na které je přístupná aplikace pro čtení a tedy i adresa, na které je přístupné webové rozhraní. Potom se zavolá metoda `Navigate`, které se jako parametr předá celá adresa pro zavolání akční metody `DetailsFromInventnum(string inventnum)` ve webovém rozhraní.

```
private void openDetail(string inventnum)
{
    String host = Application.Current.Host.Source.ToString().Substring(0, Application.Current.Host.
        Source.ToString().IndexOf("ClientBin") -1);
    HtmlPage.Window.Navigate(new Uri(host + "/Device/DetailsFromInventnum?inventnum=" +
        inventnum));
}
```

Výpis 9: Metoda openDetail volaná po přečtení QR kódu

Aby byla Silverlight aplikace přítupná ze všech stránek webové aplikace, rozhodl jsem se odkaz na ní umístit do horního baneru aplikace. Ten je definován v MasterPage View `Site.Master`. Po kliknutí na odkaz dojde k zobrazení Silverlight aplikace v dialogovém okně, což zajišťuje komponenta jQuery knihovny `dialog`. Princip je stejný, jako u potvrzovacích dialogů zobrazených při mazání přístrojů a dalších položek přes webové rozhraní.

6 Závěr

S výslednou aplikací vytvořenou v rámci této diplomové práce jsem velice spokojený, protože splňuje všechny požadavky ze zadání. Kromě samotné aplikace pro čtení jsem vytvořil také databázi systému pro inventarizaci a webové rozhraní, přes které se dá obsah databáze číst a měnit. Do vytvořeného webového rozhraní jsem aplikaci pro čtení QR kódů integroval, takže je snadno dostupná.

Jsem vděčný za získané zkušenosti s vývojem pro Silverlight, které jsem tvorbou této aplikace získal, protože je to skutečně unikátní technologie pro vytváření aplikací typu RIA. Kromě toho je vývoj aplikace pro Silverlight velmi podobný vývoji mobilních aplikací pro systém Windows Phone. Mobilní aplikace jsou dnes díky dostupnosti mobilních telefonů s operačními systémy na vzestupu a vývojáři těchto aplikací budou tedy stále žádanější. Také jsem rád, že jsem mohl navrhnout novou databázi pro systém a vyzkoušet vývoj aplikací s frameworkem ASP.NET MVC, který se dá po osvojení použít k poměrně snadnému vývoji i komplexních webových aplikací s logicky dělenou architekturou.

Vytvořený systém může být dále vylepšen například použitím autentizace pomocí SSO nebo LDAP protokolu. Tím by odpadla nutnost uchovávat v systému šifrovaná uživatelská hesla, ale především by byl systém uživatelsky přívětivější, protože by měli uživatelé stejné přihlašovací údaje, jako do jiných systémů a aplikací.

Dalším užitečným rozšířením mého systému by bylo vyvinutí mobilních aplikací pro nejpoužívanější operační systémy mobilních telefonů a tabletů. Jmenovitě pro iOS, Windows Phone a Android OS. Právě pro poslední zmíněný operační systém je již nyní vypsáno téma diplomové práce. Cílem ní je naprogramování mobilního klienta k systému pro inventarizaci. Tyto aplikace by mohly být velmi užitečné pro vyhledávání v systému pomocí QR kódů, protože mnou vyvinutá aplikace pro čtení vyžaduje plugin nainstalovaný do prohlížeče a ten je k dispozici jen pro operační systémy stolních počítačů a notebooků.

Po nasazení systému do provozu bude jistě užitečné dotvoření emailových notifikací. Například pro oznámení žádosti uživatele administrátorovi systému, který tak na ni bude moci rychleji reagovat.

Věřím, že mnou vytvořený systém najde na katedře telekomunikací VŠB-TUO své uplatnění a umožní zjednodušení inventarizace jejího majetku.

7 Reference

[1] SANSWEET, Judith. *Introducing the QR Code: the Reality & the Magic: A QR Code primer*. Auckland, N.Z: ProofreadNZ Ltd, 2011. ISBN 04-731-8451-6.

[2] ISO 18004: 2006. *Information technology — Automatic identification and data capture techniques — QR Code 2005 bar code symbology specification*. Geneva: ISO, 2006.

[3] DENSO WAVE INCORPORATED. *QRcode.com* [online]. 2013 [cit. 2013-04-04]. Dostupné z: <http://www.qrcode.com>

[4] *Zxing - Multi-format 1D/2D barcode image processing library with clients for Android, Java* [online]. 2007 [cit. 2013-04-20]. Dostupné z: <http://code.google.com/p/zxing/>

[5] Micro QR code support. In: *ZXing* [online]. 2007 [cit. 2013-04-30]. Dostupné z: <http://code.google.com/p/zxing/issues/detail?id=185>

[6] MORONEY, Laurence. *Microsoft Silverlight 4 step by step*. Sebastopol, CA: O'Reilly Media, c2010, xviii, 306 p. ISBN 07-356-3887-X.

[7] *Moonlight - Mono* [online]. 2009 [cit. 2013-04-24]. Dostupné z: <http://www.mono-project.com/Moonlight/>

[8] Databáze. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2013-04-20]. Dostupné z: <http://cs.wikipedia.org/wiki/Datab%C3%A1ze>

[9] SQL Server Express Edition. *Microsoft* [online]. 2013 [cit. 2013-04-20]. Dostupné z: <http://www.microsoft.com/express/sql/>

[10] The Repository Pattern. *MSDN - the Microsoft Developer Network* [online]. 2013 [cit. 2013-04-20]. Dostupné z: <http://msdn.microsoft.com/en-us/library/ff649690.aspx>

[11] FREEMAN, Adam a Steven SANDERSON. *Pro ASP.NET MVC 3 framework*. 3rd. ed. New York: Apress, c2011, xxv, 824 s. The expert's Voice in.NET. ISBN 978-1-4302-3404-3.

[12] ASP.NET MVC v praxi od A do Z, 6. díl – CRUD. KOTTNAUER, Jakub. *Programujte.com* [online]. 2009 [cit. 2013-04-04]. Dostupné z: <http://programujte.com/clanek/2009072800-asp-net-mvc-v-praxi-od-a-do-z-6-dil-crud/>

[13] LibreOffice. THE DOCUMENT FOUNDATION. *LGPL License » LibreOffice* [online]. 2007 [cit. 2013-15-04]. Dostupné z: <http://www.libreoffice.org/download/license>

[14] *IrfanView* [online]. 2012 [cit. 2013-04-15]. Dostupné z: <http://www.irfanview.com/>

A Uživatelská dokumentace systému pro inventarizaci

Systém je určen k inventarizaci majetku. U každého kusu majetku je evidováno jeho inventární číslo, název, název v původním jazyce, popis, výrobce a místnost, do které patří. Zároveň je k předmětu možné zadat, ve které místnosti se nacházel v minulosti, nachází nyní nebo se bude nacházet v budoucnu. K předmětům je také možné ukládat soubory a fotografie. U všech záznamů je evidováno, kdo je vložil a kdo naposledy změnil. Systém obsahuje aplikaci, které je schopna pomocí kamery v počítači přečíst QR kód umístěný na předmětu a podle něj vyhledat o předmětu všechny informace z databáze.

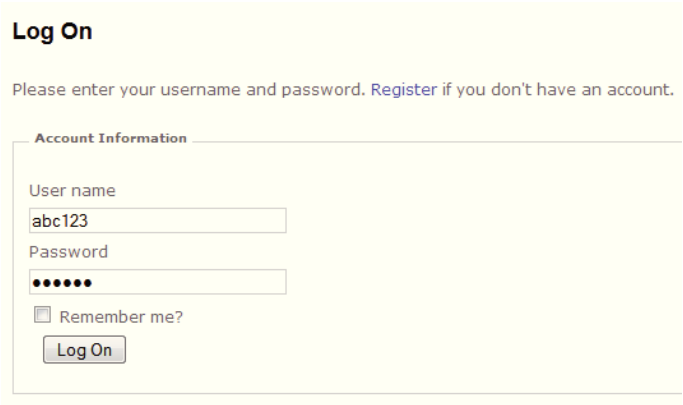
Po otevření systému se zobrazí následující stránka, která vyzývá k přihlášení. To lze provést volbou „Sign in“ v pravém horním rohu.

Obrázek 12: Úvodní stránka systému pro inventarizaci

Po kliknutí na volbu pro přihlášení je možné zadat přihlašovací údaje nebo přejít na odkaz pro registraci. Pokud se při přihlášení zadá k uživatelskému jménu pětkrát nesprávně heslo, dojde k zamknutí účtu s tímto uživatelským jménem. Odemknutí může provést pouze uživatel v roli administrátora.

Každý přihlášený uživatel bez přiřazené role má přístup ke všem informacím o majetku s výjimkou souborů (vidí jejich názvy, ale nemůže je stahovat; pro stáhnutí musí mít uživatel přiřazenu roli). Žádné z těchto informací nemůže měnit. Pro vyšší práva může mít uživatel přiřazeny následující role v libovolné kombinaci:

- uživatel zařízení (device user) – má práva přidávat, měnit a mazat aktuální umístění předmětů
- autor (author) – má práva přidávat, měnit a mazat předměty, fotografie předmětů a soubory předmětů; přidávat, měnit a mazat výrobce a místnosti
- administrátor (admin) – má práva nahlížet na smazané předměty, výrobce, místnosti, fotografie a soubory; přidávat/odebírat role uživatelů; měnit emailovou adresu uživatelů; deaktivovat/odblokovat přístup uživatelů; generovat uživatelům nové heslo pro přihlášení



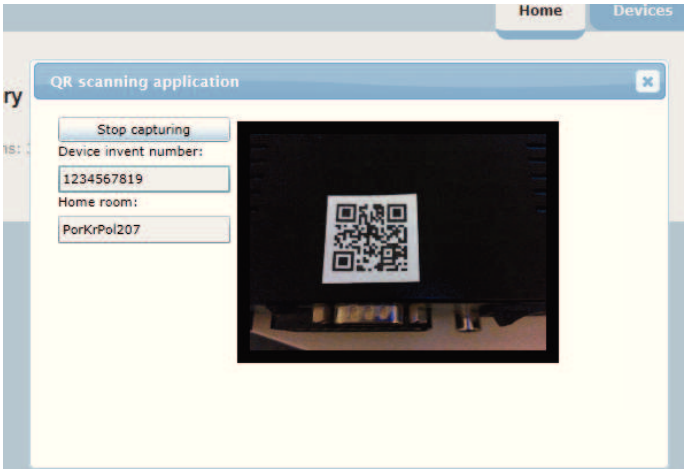
Obrázek 13: Přihlašovací obrazovka

A.1 Práva uživatele bez role

Obrázek 14: Uvítací stránka

Po přihlášení vidí uživatel na stránce uvítání a počet aktuálně evidovaných předmětů v systému. V menu má možnost přejít na předměty v systému (Devices), výrobce předmětů (Producers) a místnosti (Rooms). V pravém horním rohu se může uživatel odhlásit kliknutím na nápis „Sign out“ nebo si změnit heslo kliknutím na své přihlašovací jméno. Kliknutím na baner s nápisem „inventory system“ se uživatel z jakékoliv části systému dostane na uvítací stránku (Home). Na pravo od baneru je ikona s nápisem „SCAN QR“, kterou se v dialogovém okně otevře aplikace pro přečtení QR kódu na předmětu. Po přečtení QR kódu přesměruje aplikace uživatele na detail o předmětu, jehož inventární číslo bylo v kódu uloženo.

Na stránce s detailem může uživatel stáhnout vygenerovaný obrázek QR kódu přes odkaz pod ikonou QR kódu s nápisem „DOWNLOAD“. Stránka také zobrazuje inventární číslo předmětu, výrobce předmětu a odkaz na jeho webové stránky. Dále je vidět místnost, do které předmět patří a ve které se aktuálně nachází. Kliknutím na ikonu lupy se zobrazí



Obrázek 15: Systém se spuštěnou aplikací pro čtení QR kódu

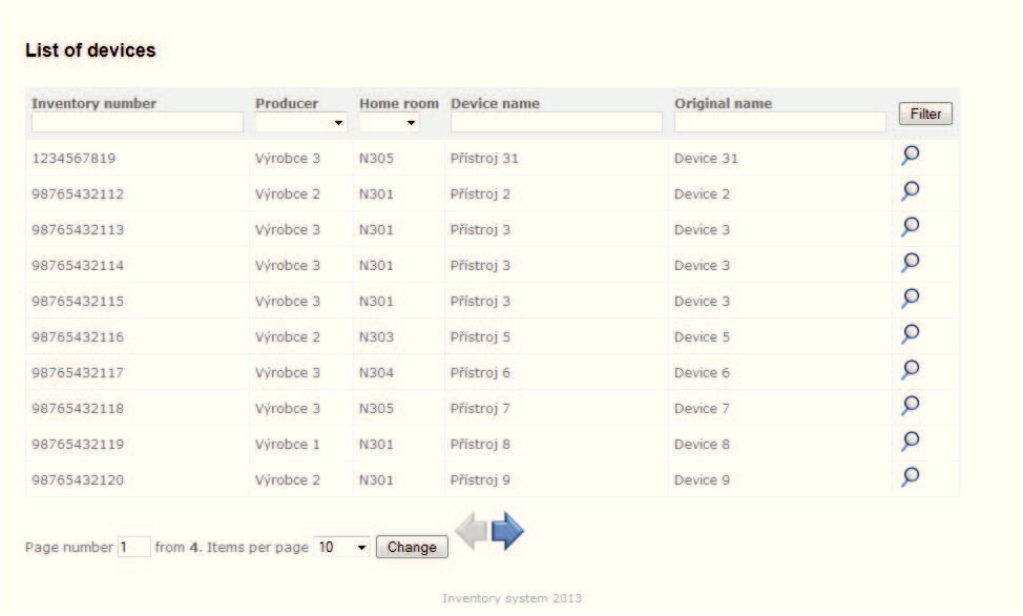
i neaktuální umístění předmětu v místnostech (minulé nebo budoucí). Na stránce se ještě nachází název předmětu, jeho název v původním jazyce, popis, fotky a soubory. Kliknutím na ikonu šipky v levém dolním rohu se načte seznam všech předmětů.

Seznam všech předmětů lze otevřít kliknutím na tlačítko „Devices“ v horním menu. Pro vyhledávání předmětů nebo zmenšení zobrazovaného obsahu lze použít filtrování v záhlaví. U inventárního čísla, názvu a původního názvu stačí zadat jejich část. Výrobce předmětu a místnost je třeba vybrat v rozbalovací nabídce.

Po načtení je vidět první stránka. Mezi stránkami je možné se pohybovat pomocí modrých šipek v zápatí. Na požadovanou stránku je možné také přejít zadáním jejího čísla do pole v zápatí. Také je zde možné změnit počet zobrazovaných předmětů na stránku.

U každé zobrazované položky lze odkazem pod ikonou lupy načíst stejný detail, jaký se zobrazí po vyhledání předmětu aplikací pro čtení QR kódu.

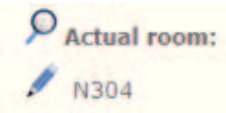
Obrázek 16: Detail předmětu



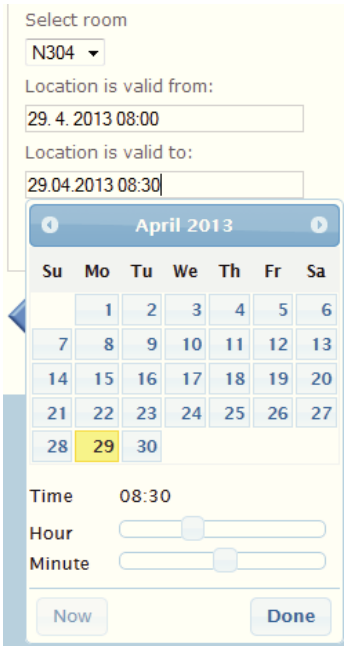
Obrázek 17: Seznam předmětů v systému

A.2 Role uživatel zařízení

Pokud má uživatel přiřazenou roli „uživatel zařízení“ (device user), má možnost měnit umístění předmětu v místnostech. K tomuto účelu má na detailu předmětu navíc ikonu tužky pro editaci aktuální místnosti. Po kliknutí na ni se načte editace aktuální místnosti. Dá se zde změnit místnost a datum platnosti záznamu o poloze předmětu. Změna data a času se provádí přímým zadáním do pole nebo vybráním z kalendáře.



Obrázek 18: Nabídka uživatele zařízení na detailu předmětu



Obrázek 19: Změna aktuální místnosti předmětu

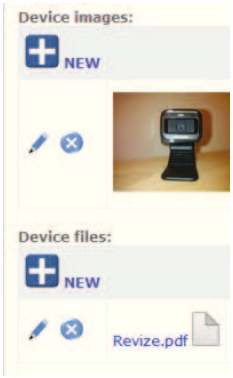
Na stránce se záznamy o umístěních předmětu může uživatel s rolí „device user“ záznamy přidávat, měnit nebo odstraňovat přes odkazy pod ikonami znaménka plus, tužky a křížku.

A.3 Role autor

Uživatelům s rolí autor přibyla na stránce s detailem předmětu ikona tužky, pod kterou je odkaz směřující na editaci předmětu. Dále mají uživatelé s touto rolí na detailu předmětu

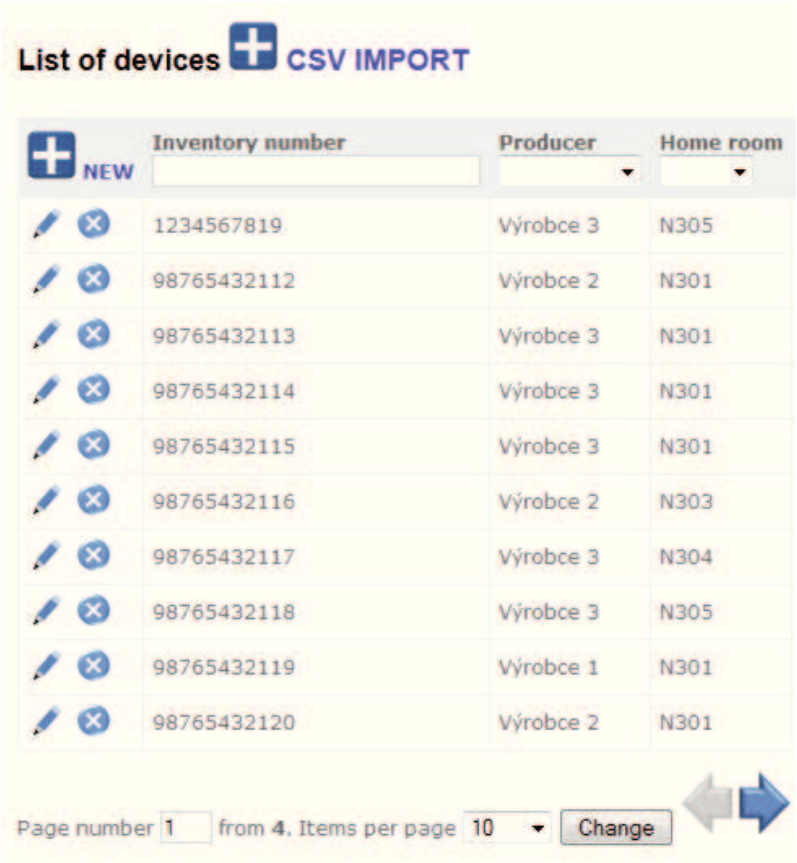
Obrázek 20: Seznam umístění předmětu

odkazy pro přidání, změnu a odstranění fotek či souborů přístroje. S rolí autor se rozšiřuje i výpis předmětů. Konkrétně o volby pro přidání předmětů, jejich editaci a také mazání.



Obrázek 21: Editace souborů a fotek

V záhlaví je také volba „CSV IMPORT“, přes kterou je možné do systému importovat záznamy o předmětech z csv souboru. Po kliknutí na odkaz pod touto ikonou se načte stránka, kde je možné si uložit šablonu csv souboru. Po vyplnění souboru a jeho uložení v počítači jej stačí na stránce vybrat přes tlačítko „Procházet“ a nahrát ho. Podle inventárního čísla se v systému zkontroluje existence předmětu. Pokud neexistuje, tak se vloží. Pokud ale s tímto inventárním číslem existuje už jiný předmět, je záznam z csv souboru ignorován. Podle názvu výrobce a místnosti v csv souboru se vyhledá jejich existence v číselníku výrobců a místností. Pokud existují, přiřadí se importovanému záznamu již existující výrobce či místnost. Pokud neexistují, tak se založí do číselníku jako nový výrobce nebo nová místnost.



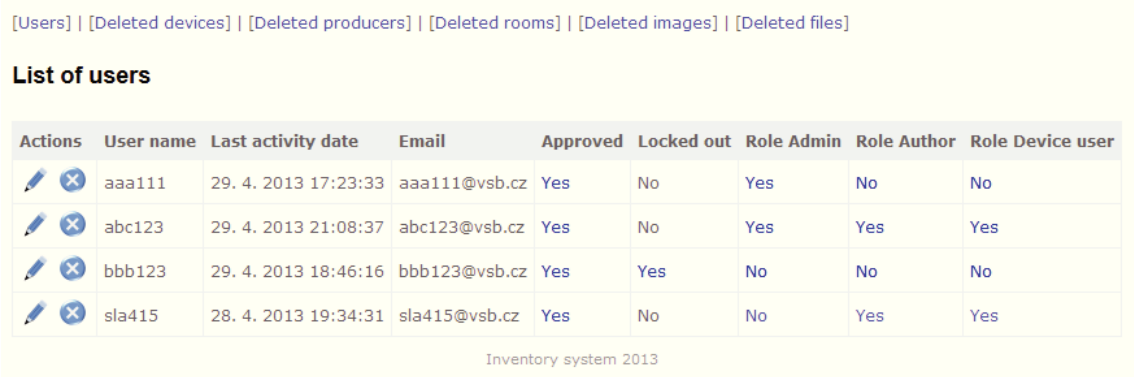
Obrázek 22: Výpis předmětů pod rolí autor

Obrázek 23: Import z csv souboru

A.4 Role administrátor

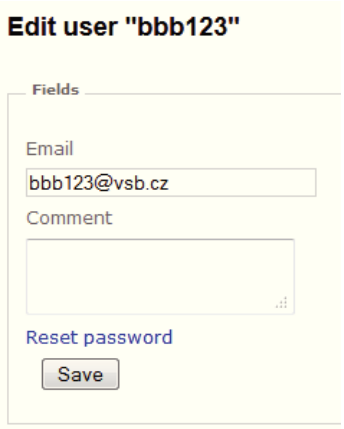
S touto rolí přibývá v hlavní nabídce záložka „Admin“ a na ní je k dispozici další podnabídka. Kliknutím na volbu „Users“ se načte seznam všech registrovaných uživatelů.

V seznamu je jim možné schválit nebo zamítnout přístup pomocí kliknutí na hodnotu sloupce „Approved“. Pokud je jeho hodnota „Yes“, může se uživatel přihlásit, ale pokud je „No“, přihlášení se uživateli nepodaří. Když uživatel zadá heslo při přihlášení pětkrát nesprávně, účet se mu zablokuje a ve sloupci „Locked out“ uvidí administrátor u tohoto uživatele hodnotu „Yes“. Kliknutím na na tuto hodnotu účet uživateli odemkne. Poslední tři sloupce zobrazují role přiřazené uživatelům a umožňují kliknutím jejich změnu.



Obrázek 24: Administrace uživatelů

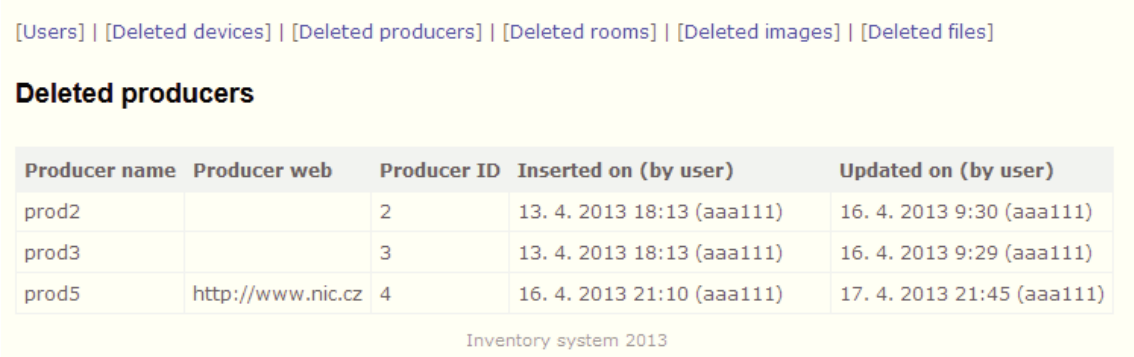
V editaci uživatele může administrátor změnit jeho emailovou adresu nebo kliknutím na tlačítko „Reset password“ uživateli resetovat heslo. Po kliknutí na odkaz pro resetování se nově vygenerované heslo okamžitě objeví na místo původního textu odkazu. Pokud byla editace otevřena jen kvůli resetu hesla, není třeba změny ukládat tlačítkem Save.



Obrázek 25: Editace uživatele

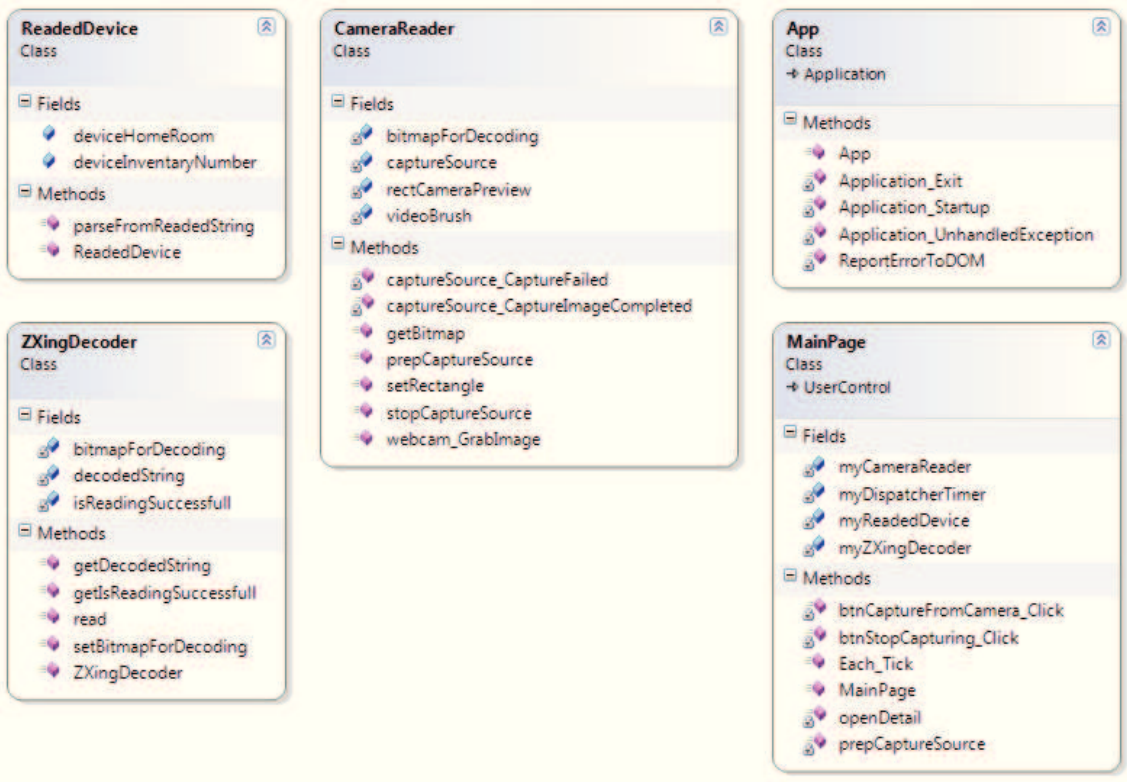
Administrátor má v dalších volbách své podnabídky přístup k již smazaným záznamům o předmětech, výrobcích, místnostech, fotkách a souborech. Ty se již nedají v systému obnovit, ale přes tuto volbu se o těchto položkách dají zjistit původní infor-

mace, pokud to je potřeba. Fotky a soubory již smazaných předmětů se dají přes tuto volbu stáhnout ze systému na disk.



Obrázek 26: Smazané záznamy výrobců

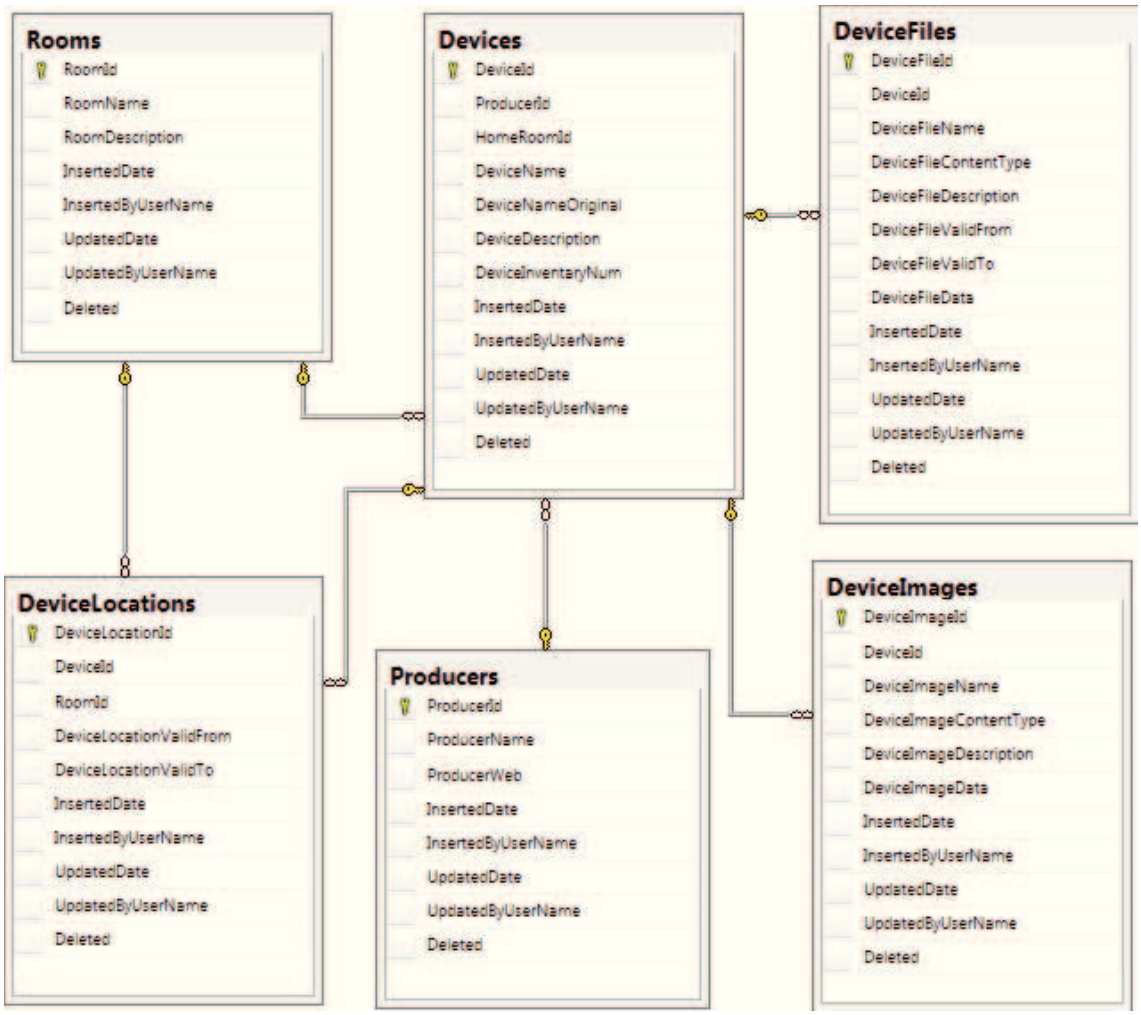
B Třídní diagram aplikace pro čtení QR kódu



Obrázek 27: Třídní diagram aplikace pro čtení QR kódu

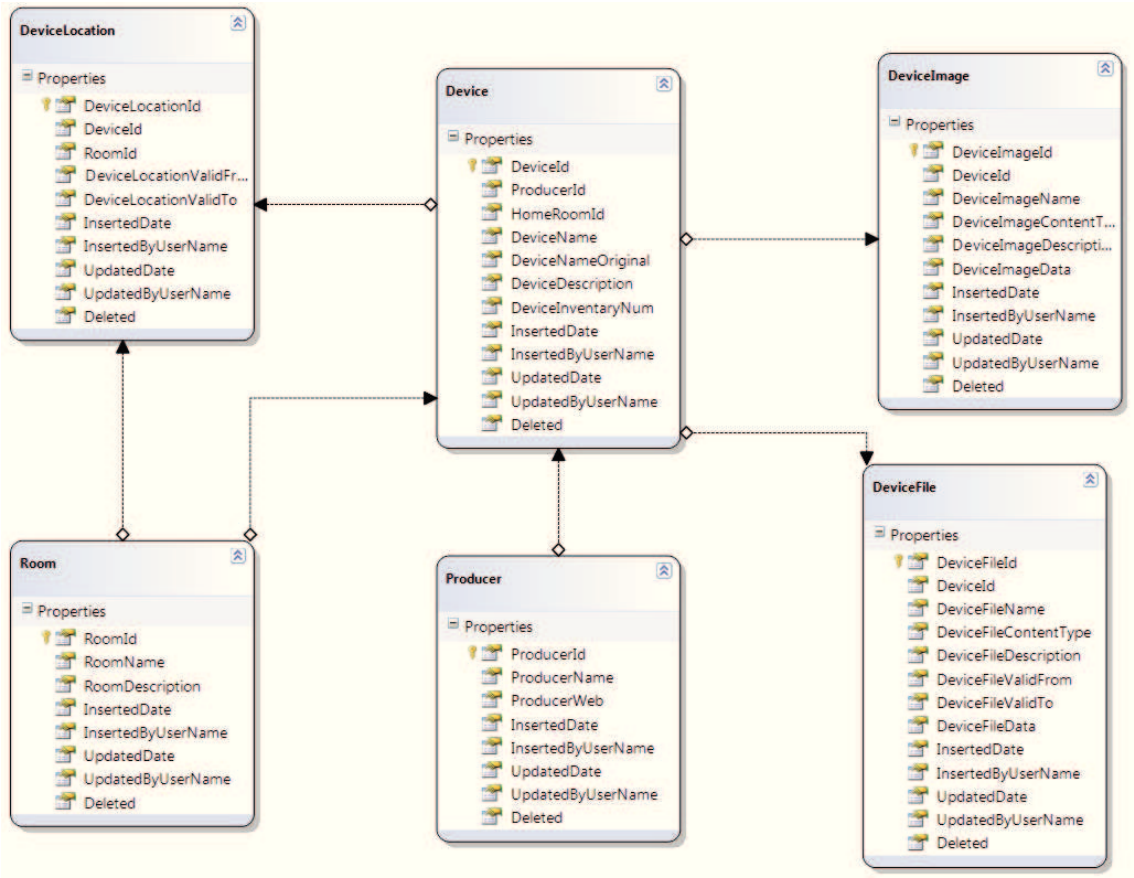
C Databáze systému

C.1 Struktura databáze



Obrázek 28: Struktura databáze pro systém

C.2 Třídní diagram datového modelu



Obrázek 29: Třídní diagram datového modelu v systému

C.3 Atributy databázových tabulek

název atributu	**datový typ**	**popis**
DeviceId (PK)	int	identifikátor přístroje
ProducerId (FK)	int	identifikátor výrobce
RoomId (FK)	int	identifikátor místnosti
DeviceName	nvarchar(256)	název přístroje
DeviceNameOriginal	nvarchar(256)	název přístroje v původním jazyce
DeviceDescription	nvarchar(MAX)	popis přístroje
DeviceInventaryNum	nvarchar(256)	inventární číslo přístroje
InsertedDate	datetime	datum a čas vložení záznamu
InsertedByUserName	nvarchar(256)	uživatel, který vložil záznam
UpdatedDate	datetime	datum a čas poslední aktualizace záznamu
UpdatedByUserName	nvarchar(256)	uživatel, který naposledy změnil záznam
Deleted	bit	1 = smazaný přístroj, 0 = platný přístroj

Tabulka 3: Databázová tabulka přístrojů (Devices)

název atributu	**datový typ**	**popis**
DeviceFileId (PK)	int	identifikátor souboru přístroje
DeviceId (FK)	int	identifikátor přístroje
DeviceFileName	nvarchar(256)	jméno souboru
DeviceFileContentType	nvarchar(256)	typ formátu souboru
DeviceFileDescription	nvarchar(MAX)	popis souboru
DeviceFileData	varbinary(MAX)	data souboru
DeviceFileValidFrom	datetime	datum a čas platnosti záznamu
DeviceFileValidTo	datetime	datum a čas platnosti záznamu
InsertedDate	datetime	datum a čas vložení záznamu
InsertedByUserName	nvarchar(256)	uživatel, který vložil záznam
UpdatedDate	datetime	datum a čas poslední aktualizace záznamu
UpdatedByUserName	nvarchar(256)	uživatel, který naposledy změnil záznam
Deleted	bit	1 = smazaný soubor, 0 = platný soubor

Tabulka 4: Databázová tabulka souborů přístroje (DeviceFiles)

název atributu	**datový typ**	**popis**
DeviceImageId (PK)	int	identifikátor fotografie přístroje
DeviceId (FK)	int	identifikátor přístroje
DeviceImageName	nvarchar(256)	jméno souboru
DeviceImageContentType	nvarchar(256)	typ formátu souboru
DeviceImageDescription	nvarchar(MAX)	popis fotografie
DeviceImageData	varbinary(MAX)	data souboru s fotografií
InsertedDate	datetime	datum a čas vložení záznamu
InsertedByUserName	nvarchar(256)	uživatel, který vložil záznam
UpdatedDate	datetime	datum a čas poslední aktualizace záznamu
UpdatedByUserName	nvarchar(256)	uživatel, který naposledy změnil záznam
Deleted	bit	1 = smazaná fotografie, 0 = platná fotografie

Tabulka 5: Databázová tabulka fotografií přístroje (DeviceImages)

název atributu	**datový typ**	**popis**
RoomId (PK)	int	identifikátor místnosti
RoomName	nvarchar(256)	název místnosti
RoomDescription	nvarchar(MAX)	popis místnosti
InsertedDate	datetime	datum a čas vložení záznamu
InsertedByUserName	nvarchar(256)	uživatel, který vložil záznam
UpdatedDate	datetime	datum a čas poslední aktualizace záznamu
UpdatedByUserName	nvarchar(256)	uživatel, který naposledy změnil záznam
Deleted	bit	1 = smazaná místnost, 0 = platná místnost

Tabulka 6: Databázová tabulka místností (Rooms)

název atributu	**datový typ**	**popis**
ProducerId (PK)	int	identifikátor výrobce
ProducerName	nvarchar(256)	název výrobce
ProducerWeb	nvarchar(256)	webová adresa výrobce
InsertedDate	datetime	datum a čas vložení záznamu
InsertedByUserName	nvarchar(256)	uživatel, který vložil záznam
UpdatedDate	datetime	datum a čas poslední aktualizace záznamu
UpdatedByUserName	nvarchar(256)	uživatel, který naposledy změnil záznam
Deleted	bit	1 = smazaný výrobce, 0 = platný výrobce

Tabulka 7: Databázová tabulka výrobců (Producers)

název atributu	**datový typ**	**popis**
DeviceLocationId (PK)	int	identifikátor pozice přístroje
DeviceId (FK)	int	identifikátor přístroje
RoomId (FK)	int	identifikátor místnosti
DeviceLocationValidFrom	datetime	datum a čas platnosti záznamu
DeviceLocationValidTo	datetime	datum a čas platnosti záznamu
InsertedDate	datetime	datum a čas vložení záznamu
InsertedByUserName	nvarchar(256)	uživatel, který vložil záznam
UpdatedDate	datetime	datum a čas poslední aktualizace záznamu
UpdatedByUserName	nvarchar(256)	uživatel, který naposledy změnil záznam
Deleted	bit	1 = smazaná pozice, 0 = platná pozice

Tabulka 8: Databázová tabulka aktuálních pozic přístroje (DeviceLocations)